

A Babel language definition file for French

frenchb.dtx v3.6b, 2024-06-26

Daniel Flipo
daniel.flipo@free.fr

Contents

1	The French language	2
1.1	Basic interface	2
1.2	Customisation	5
1.2.1	<code>\frenchsetup</code>	5
1.2.2	Caption names	10
1.2.3	Figure and table captions	10
1.3	Hyphenation checks	11
1.4	Changes	12
2	The code	14
2.1	Initial setup	14
2.2	Punctuation	17
2.2.1	Punctuation with LuaTeX	21
2.2.2	Punctuation with XeTeX	32
2.2.3	Punctuation with standard (pdf)TeX	36
2.2.4	Punctuation switches common to all engines	38
2.3	Commands for French quotation marks	39
2.4	Date in French	44
2.5	Extra utilities	45
2.6	Formatting numbers	49
2.7	Caption names	52
2.8	Figure and table captions	55
2.9	Dots...	57
2.10	More checks about packages' loading order	58
2.11	Setup options: key/value stuff (l3keys)	59
2.12	French lists	74
2.13	French indentation of sections	80
2.14	Formatting footnotes	80
2.15	Clean up and exit	85
2.16	Files <code>frenchb.ldf</code> , <code>francais.ldf</code> , <code>canadien.ldf</code> and <code>acadian.ldf</code>	85
3	Change History	88

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.

`babel-french` has been improved using helpful suggestions from many people, mainly from Jacques André, Michel Bovani, Thierry Bouche, Vincent Jalby, Denis Bitouzé, Ulrike Fisher and Marcel Krüger. Thanks to all of them!

LaTeX-2.09 is no longer supported. Version 3.0 has been designed to be used only with LaTeX2e and Plain formats based on TeX, pdfTeX, LuaTeX or XeTeX engines.

Changes between version 3.0 and v3.6b are listed in subsection 1.4 p. 12.

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

A variant `acadian` of `french` is provided; it is originally identical to `french` but can be customised independently in terms of patterns, punctuation spacing, captions, etc. Both variants can be used together inside the same document.

`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);
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);

¹The file described in this section has version number v3.6b and was last revised on 2024-06-26.

²Always 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).

3. vertical spacing in general LaTeX lists is shortened;
4. footnotes are displayed “à la française”.
5. the separator following the table or figure number in captions is printed as ‘ – ’ instead of ‘: ’; for changing this see [1.2.3 p. 10](#).

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 spacing⁵ in French; this is achieved using callbacks in Lua(La)TeX or ‘XeTeXinterchar’ mechanism in Xe(La)TeX; with TeX’82 and pdf(La)TeX these four characters are made active in the whole document;
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. 10](#).
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, to get proper spacing in LaTeX2e see option `og=«, fg=»` p. 8.

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. 9.

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.

- with all engines: the inner quotation is surrounded by double quotes (“*texte*”) unless option `InnerGuillSingle=true`, then a) the inner quotation is printed as `< texte >` and b) if the inner quotation spreads over more than one paragraph, every paragraph included in the inner quotation starts with a `<` or a `>` or nothing, depending on option `EveryParGuill=open` (default) or `=close` or `=none`.

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

⁵Well, the automatic insertion may add unwanted spaces in some cases, for correction see `AutoSpacePunctuation` option and `\NoAutoSpacing` command p. 7.

- with LuaTeX based engines, 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` so that `\frquote{}` behaves as with non-LuaTeX engines.

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 are provided to typeset the symbol for “degré”: `\degre` prints the raw character and `\degres` should be used to typeset temperatures (e.g., “20~\degres C” with a non-breaking space), or for alcohols’ strengths (e.g., “45\degres” with *no* space in French) or for angles in math mode.
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 `'1er juin'` (no need for a forced space after `1\ier`).

1.2 Customisation

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

1.2.1 `\frenchsetup{options}`

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

The other options are listed below. Their default value is shown between braces, sometimes followed 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.

`GlobalLayoutFrench=false` (`true*`) can only be used when French is the main language; setting it to `false` will emulate what prior versions of `babel-french` (pre-2.2) did: lists, and first paragraphs of sections will be displayed the standard way in other languages than French, and “à la française” in French (changing the layout inside a document is a bad practice imho). Note that the layout of footnotes is language independent anyway (see below `FrenchFootnotes` and `AutoSpaceFootnotes`).

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”.

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

Text starting at ‘parindent’ <= Leftmargin – first item running on two lines or more... – first second level item on two lines... – next one... – second item...	Text starting at ‘parindent’ <= Leftmargin – first item running on two lines or more... – first second level item on two lines... – next one... – second item...
Default French layout	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`.

`ListOldLayout=true (false)`; starting with version 2.6a, the layout of lists has changed regarding leftmargins' sizes and default `itemize` label ('—' instead of '–' up to 2.5k). This option, provided for backward compatibility, displays lists as they were up to version 2.5k.

`FrenchFootnotes=false (true*)` reverts to the standard layout of footnotes. By default `babel-french` typesets leading numbers as '1. ' instead of '1', but has no effect on footnotes numbered with symbols (as in the `\thanks` command). Two commands `\StandardFootnotes` and `\FrenchFootnotes` are available to change the layout of footnotes locally; `\StandardFootnotes` can help when some footnotes are numbered with letters (inside `minipages` for instance).

`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)`; in French, the user *should* input a space before the four characters ' : ; ! ? ' but as many people forget about it (even among native French writers!), the default behaviour of `babel-french` is to automatically typeset non-breaking spaces the width of which is either `\FBthinSPACE` (defaults to a thin space) before ' ; ' ! ' ? ' or `\FBcolonspace` (defaults to `\space`) before ' : ' ; the defaults follow the French 'Imprimerie Nationale's recommendations. This is convenient in most cases but can lead to addition of spurious spaces in URLs, in MS-DOS paths or in timetables (10:55) —this no longer occurs with LuaTeX—, except if they are typed in `\texttt` or `verbatim` mode. When the current font is a monospaced (typewriter) font, no spurious space is

added in that case ⁷, so the default behaviour of `babel-french` in that area should be fine in most circumstances.

Choosing `AutoSpacePunctuation=false` will ensure that a proper space is added before ‘:;!?’ *if and only if* a (normal) space has been typed in. This option gives full control on space insertion before ‘:;!?’ . Those who are unsure about their typing in this area should stick to the default option and use the provided `\NoAutoSpacing` command inside a group in case an unwanted space is added by `babel-french` (i.e. `{\NoAutoSpacing http://mysite}` ⁸ or `{\NoAutoSpacing ???}` (needed for pdfTeX only).

`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=»`; when guillemets characters are available on the keyboard (through a compose key for instance), it is nice to use them instead of typing `\frquote{}`. This option tells `babel-french` which characters are opening and closing French guillemets (they depend on the input encoding), then you can type either `« guillemets »` or `«guillemets»` ⁹ (with or without spaces) to get properly typeset French quotes. This option works with LuaLaTeX, XeLaTeX and with pdfLaTeX (default encoding: utf8); with pdfLatex other 8-bits encodings (latin1, latin9, ansinew, applemac,...) are also supported when properly declared with `inputenc`.

`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.

⁷Unless option `OriginalTypewriter` is set, `\ttfamily` is redefined in French to switch off space tuning, see below.

⁸Actually, this is needed only with the XeTeX and pdfTeX engines. LuaTeX no longer inserts any space in strings like `http://mysite`, `C:\Foo`, `10:55...`

⁹Or even `«~guillemets~»`, but *only* with LuaLaTeX.

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 (~)¹⁰ 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=false (true*); when set to **false** the default separator (colon) is used instead of `\CaptionSeparator`. Anyway, `babel-french` tries hard to insert a proper space before it in French and warns if it fails to do so.

OldFigTabCaptions=true (false) is to be used *only* when figures’ and tables’ captions must be typeset as with pre 3.0 versions of `babel-french` (with `\CaptionSeparator` in French and colon otherwise). Intended for standard LaTeX classes only.

FrenchSuperscripts=false (true); then `\up=\textsuperscript`. (option added in version 2.1). Should only be made **false** to recompile documents written before 2008 without changes: by default `\up` now relies on `\fup` designed to produce better looking superscripts.

¹⁰Actually without stretch nor shrink.

`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}` or `\def\acadianproofname{Preuve}` for the acadian dialect. The older syntax `\addto\captionsfrench{\def\proofname{Preuve}}` still works. Keep in mind that *only* `french` can be used to redefine captions, even if `Babel`'s option was entered as `frenchb` or `francais`.

1.2.3 Figure and table captions

In French, captions in figures and tables should never be printed as 'Figure 1: ' which is the default in standard `LaTeX2e` classes (a space should *always* precede a colon in French), anyway 'Figure 1 - ' is preferred.

When French is the main language, the default behaviour of `babel-french` is to change the separator (colon) used in figures' and tables' captions *for all languages* to `\CaptionSeparator` which defaults to ' - ' and can be redefined in the preamble with `\renewcommand*{\CaptionSeparator}{...}`. This works for the standard `LaTeX2e` classes, for the `memoir` `koma-script` and `beamer` classes. In case this procedure fails a warning is issued.

When French is not the main language, the colon is preserved for all languages including French but `babel-french` tries hard to insert a proper space before it and warns if it fails to do so.

Three options are provided to customise figure and table captions:

- `CustomiseFigTabCaptions` is set to `true` when French is the main language (hence separator = ' - ') and to `false` otherwise (hence separator = ': ' with a proper space before the colon in French if possible); toggle this option if needed;

- the second option, `OldFigTabCaptions`, can be set to `true` to print figures' and tables' captions as they were with versions pre 3.0 of `babel-french` (using `\CaptionSeparator` in French and colon in other languages); this option only makes sense with the standard LaTeX classes `article`, `report` and `book`;
- the last option, `SmallCapsFigTabCaptions`, 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 pdfLaTeX on the following file:

```
%%% Test file for French hyphenation.
\documentclass[french]{article}
\usepackage[utf8]{inputenc} % utf8, what else?
\usepackage[T1]{fontenc}    % mandatory for French
\usepackage{lmodern}       % or erewhon, palatino...
\usepackage{babel}
\begin{document}
\showhyphens{signal container \`ev\`enement alg\`ebre}
\showhyphens{signal container événement algèbre}
\end{document}
```

- check the hyphenations proposed by T_EX in your log-file; in French you should get with both 7-bit and 8-bit encodings
`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).

Frequent mismatches:

- you get `sig-nal con-tainer`, this probably means that the hyphenation patterns you are using are for US-English, not for French;
- you get no hyphen at all in `évé-ne-ment`, this probably means that you are using CM fonts and the macro `\accent` to produce accented characters. Using 8-bits fonts with built-in accented characters avoids this kind of mismatch.

1.4 Changes

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`.

What's new in version 3.5?

Version 3.5a offers a new option `ListItemsAsPar`. The default layout of lists is unchanged (for backward compatibility), but users should try this new option which ensures a layout of lists closer to French typographic standards: see f.i. how lists are typeset in the book "Lexique des règles typographiques en usage à l'Imprimerie Nationale".

Version 3.5b fixes a bug due to wrong `\everypar`'s management in `\frquote{}`; it showed up when `\frquote{}` immediately followed a sectioning command.

Starting with version 3.5d, a new option `StandardListSpacing` has been added to supersede `ReduceListSpacing`.

A new command `\NoEveryParQuote` has been added in version 3.5e: it is meant to be used inside a group or environment to suppress unwanted guillemets (typically when lists are embedded in `\frquote{}`).

Version 3.5g fixes a long standing bug affecting LuaTeX: legacy kerning was disabled for Type1 fonts since v3.1g (2015).

Version 3.5j also fixes a long standing bug affecting koma-script, `memoir` and `beamer` classes: redefinitions of the caption separator (commands `\captionformat`, `\captiondelim`, etc.) are now taken into account properly.

Version 3.5k is a cleanup release:

- the translations in French of `\figurename` and `\tablename` no longer hold font changing commands (switch to small caps), the font switch has been moved to `\fnum@figure` and `\fnum@table` as suggested by Axel Sommerfeldt.
- Package `caption` can now be loaded whether before or after `babel`, indifferently.
- `\pdfstringdefDisableCommands` is no longer used: as suggested by the LaTeX3 team, all commands requiring special care in `hyperref`'s bookmarks are now defined using `\textorpdfstring{ }{ }`.

Version 3.5n introduces a new command `\bname{}` (an alternative to `\bsc{}`).

Version 3.5q corrects a bug in lists layout: `\listparindent` (formerly `0pt`) is defined as `\parindent` and if `\parskip > 0pt`, `\parsep` is now defined as `\parskip`. This ensures that paragraphs included in lists are now visible. The former behaviour can be recovered by adding `\parskip=0pt, \parindent=0pt` inside the list environment.

Version 3.5r is compatible with `ucharclasses` which is now loaded by `fontsetup` with the XeTeX engine. The `frenchb.ins` file is no longer needed to extract the `.ldf` files from `frenchb.dtx` (see `README.md`).

What's new in version 3.4?

Version 3.4a adds a new command `\frenchdate` (see p. 4) and slightly changes number formatting: `\FBthousandsep` is now a *kern* instead of a rubber length. `\renewcommand*\FBthousandsep{~}` will switch back to the former (wrong) behaviour.

Both options `french` and `acadian` can now be used simultaneously in a document; currently `french` and `acadian` are identical, it is up to the user to customise `acadian` in terms of hyphenation patterns, captionnames, date format or high punctuation and quotes spacing if he/she needs a variant for French.

A new command `\FBsetspaces` has been added for easy customising of spacing before high punctuation and inside quotes independently for `french` and `acadian`, see p. 18.

Version 3.4 requires eTeX and LuaTeX 1.0.4 or newer.

What's new in version 3.3?

In version 3.3d the automatic insertion of non-breaking spaces before the colon character has been improved *with engine LuaTeX only*: a spurious space is no longer inserted in strings like `http://mysite, C:\Program Files or 10:55`. Unfortunately, my attempts to do the same with XeTeX or pdfTeX were unsuccessful.

A few internal changes have been made in version 3.3c to improve the conversion into HTML of non-breaking spaces added by `babel-french`. Usage of `lwrap` (v.0.37 and up) is recommended for HTML output, it works fine on files compiled with XeLaTeX or pdfLaTeX formats. A new experimental option `UnicodeNoBreakSpaces` has been added for LuaLaTeX in version 3.3c, see p. 8.

According to current Babel's standards, every dialect should have its own `.ldf` file; starting with version 3.3b, the main support for French is in `french.ldf`, portman-teau files `frenchb.ldf`, `français.ldf`, `acadian.ldf` and `canadien.ldf` have been added. Recommended options are `french` or `acadian`, all other are deprecated. BTW, options `french` and `acadian` are currently strictly identical.

Release 3.3a is compatible with LuaTeX v. 0.95 (TL2016) and up. Former skips `\FBcolonskip`, `\FBthinskip` and `\FBguillskip` controlling punctuation spacings in LuaTeX have been removed; all three engines now rely on the same commands `\FBcolonspace`, `\FBthinspace` and `\FBguillspace`.

An alias `\frenchsetup{}` for `\frenchbsetup{}` has been added in version 3.3a, it might appear more relevant in the future as the language name `frenchb` should vanish.

Further customisation of the `\part{}` command is provided via three new commands `\frenchpartfirst`, `\frenchpartsecond` and `\frenchpartnameord`.

2 The code

2.1 Initial setup

The macro `\LdfInit` takes care of preventing that this file is loaded more than once (even if both options `french` and `acadian` are used in the same document), checking the category code of the `@` sign, etc.

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

Let's provide a substitute for `\PackageError`, `\PackageWarning` and `\PackageInfo` not defined in Plain:

```
3 \def\fb@error#1#2{%
4   \begingroup
5     \newlinechar=`^^J
6     \def\{\^^J(french.ldf) }%
7     \errhelp{#2}\errmessage{\#1^^J}%
8   \endgroup}
9 \def\fb@warning#1{%
10  \begingroup
11    \newlinechar=`^^J
12    \def\{\^^J(french.ldf) }%
13    \message{\#1^^J}%
14  \endgroup}
15 \def\fb@info#1{%
16  \begingroup
17    \newlinechar=`^^J
18    \def\{\^^J}%
19    \wlog{#1}%
20  \endgroup}
```

Quit if eTeX is not available.

```
21 \let\bbl@tempa\relax
22 \begingroup\expandafter\expandafter\expandafter\endgroup
23 \expandafter\ifx\csname eTeXversion\endcsname\relax
24   \let\bbl@tempa\endinput
25   \fb@error{babel-french requires eTeX.\\
26             Aborting here}
27             {Original PlainTeX is not supported,\\
28             please use LuaTeX or XeTeX engines.}
29 \fi
30 \bbl@tempa
```

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

```
31 \let\bbl@tempa\relax
32 \ifdefined\babeltags
33 \else
```

```

34 \let\bb1@tempa\endinput
35 \ifdefined\PackageError
36   \PackageError{french.ldf}
37     {babel-french requires babel v.3.16.\MessageBreak
38     Aborting here}
39     {Please upgrade Babel!}
40 \else
41   \fb@error{babel-french requires babel v.3.16.\\
42     Aborting here}
43     {Please upgrade Babel!}
44 \fi
45 \fi
46 \bb1@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>`.

```

47 \def\FB@nopatterns{%
48   \ifdefined\l@nohyphenation
49     \addialect\l@french\l@nohyphenation
50     \edef\bb1@nulllanguage{\string\language=nohyphenation}%
51   \else
52     \edef\bb1@nulllanguage{\string\language=0}%
53     \addialect\l@french0
54   \fi
55   \@nopatterns{French}}
56 \ifdefined\l@french \else \FB@nopatterns \fi

```

Babel's French language can be loaded with option `acadian` which stands for Canadian French. If no specific hyphenation patterns are available, Canadian French will use the French ones.

```

57 \ifdefined\l@acadian
58   \addialect\l@canadien\l@acadian
59 \else
60   \addialect\l@acadian\l@french
61   \addialect\l@canadien\l@french
62 \fi

```

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

```

63 \providehyphenmins{french}{\tw@\thr@@}
64 \providehyphenmins{acadian}{\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.

```

65 \newif\ifLaTeXe
66 \let\bb1@tempa\relax

```



```

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

```

`\ifFBunicode` French hyphenation patterns are now coded in Unicode, see file `hyph-fr.tex`. XeTeX and LuaTeX engines require some extra code to deal with the French “apostrophe”.

`\ifBLaTeX` and `\ifBTeX` Let’s define three new ‘if’: `\ifBLaTeX`, `\ifBTeX` and `\ifFBunicode` which will be true for XeTeX and LuaTeX engines and false for 8-bits engines.

```

80 \newif\ifFBunicode
81 \newif\ifBLaTeX
82 \newif\ifBTeX
83 \begingroup\expandafter\expandafter\expandafter\endgroup
84 \expandafter\ifx\csname luatexversion\endcsname\relax
85 \else
86   \FBunicodetrue \BLaTeXtrue
87 \fi
88 \begingroup\expandafter\expandafter\expandafter\endgroup
89 \expandafter\ifx\csname XeTeXrevision\endcsname\relax
90 \else
91   \FBunicodetrue \BTeXtrue
92 \fi

```

`\ifBFfrench` True when the current language is French or any of its dialects; will be set to true by `\extrasfrench` and to false by `\noextrasfrench`. Used in `\DecimalMathComma` and `frenchsetup{og=«, fg=»}`.

```

93 \newif\ifBFfrench

```

`\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.

The following code ensures correct hyphenation of words like *d’aventure*, *l’utopie*, with all TeX engines (XeTeX, LuaTeX, pdfTeX) using `hyph-fr.tex` patterns.

```

94 \def\extrasfrench{%
95   \FBfrenchtrue
96   \babel@savevariable{\lccode"27}%
97   \lccode"27="27
98   \ifFBunicode
99     \babel@savevariable{\lccode"2019}%
100    \lccode"2019="2019
101   \fi
102 }
103 \def\noextrasfrench{\FBfrenchfalse}

```

One more thing `\extrasfrench` needs to do is to make sure that “Frenchspacing” is in effect. `\noextrasfrench` will switch “Frenchspacing” off again if necessary.

```

104 \addto\extrasfrench{\bbl@frenchspacing}
105 \addto\noextrasfrench{\bbl@nonfrenchspacing}

```

2.2 Punctuation

As long as no better solution is available, the ‘high punctuation’ characters (`;` `!` `?` and `:`) have to be made `\active` for an automatic control of the amount of space to be inserted before them. Both XeTeX and LuaTeX provide an alternative to active characters (‘XeTeXinterchar’ mechanism and LuaTeX’s callbacks).

`\ifFB@active@punct` Three internal flags are needed for the three different techniques used for ‘high punctuation’ management.

```

106 \newif\ifFB@active@punct \FB@active@puncttrue

```

`\ifFB@luatex@punct` With LuaTeX, starting with version 1.0.4, callbacks are used to get rid of active punctuation. With previous versions, ‘high punctuation’ characters remain active (see below).

```

107 \newif\ifFB@luatex@punct
108 \ifB LuaTeX
109   \ifnum\luatexversion<100
110     \ifx\PackageWarning\@undefined
111       \fb@warning{Please upgrade LuaTeX to version 1.0.4 or above!\%
112         babel-french will make high punctuation characters (;!?)\%
113         active with LuaTeX < 1.0.4.}%
114     \else
115       \PackageWarning{french.ldf}{Please upgrade LuaTeX
116         to version 1.0.4 or above!\MessageBreak
117         babel-french will make high punctuation characters%
118         \MessageBreak (;!?) active with LuaTeX < 1.0.4;%
119         \MessageBreak reported}%

```

```

120   \fi
121   \else
122     \FB@luatex@puncttrue\FB@active@punctfalse
123   \fi
124 \fi

```

`\ifFB@xetex@punct` For XeTeX, the availability of `\XeTeXinterchartokenstate` decides whether the ‘high punctuation’ characters (; ! ? and :) have to be made `\active` or not. The number of available character classes has been increased from 256 to 4096 in XeTeX v. 0.99994, the class for non-characters is now 0xFFF=4095 (formerly 0xFF=255). The class for standard characters is 0.

```

125 \newcount\FB@stdchar
126 \newif\ifFB@xetex@punct
127 \ifdefined\XeTeXinterchartokenstate
128   \FB@xetex@puncttrue\FB@active@punctfalse
129   \ifdim\the\XeTeXversion\XeTeXrevision\p@ < 0.99994\p@
130     \chardef\FB@nonchar="FF \relax
131   \else
132     \chardef\FB@nonchar="FFF \relax
133   \fi
134   \FB@stdchar=\z@
135 \fi

```

`\FBguillspace` These three commands are meant for basic French. Other French dialects can use `\FBcolonspace` different settings, see below. According to the I.N. specifications, the ‘:’ requires `\FBthinspace` an inter-word space before it, the other three require just a thin space. We define `\FBcolonspace` as `\space` (inter-word space) and `\FBthinspace` as an half inter-word space with no shrink nor stretch. `\FBguillspace` is defined btw. as spacing for French quotes is handled together with high punctuation for LuaTeX and XeTeX. `\FBguillspace` 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. A penalty will be added before these spaces to prevent line breaking.

```

136 \newcommand*{\FBguillspace}{\hskip .8\fontdimen2\font
137                               plus .3\fontdimen3\font
138                               minus .8\fontdimen4\font \relax}
139 \newcommand*{\FBcolonspace}{\space}
140 \newcommand*{\FBthinspace}{\hskip .5\fontdimen2\font \relax}

```

`\FBsetspaces` This command makes it easy to fine tune `\FBguillspace`, `\FBcolonspace` and `\FBthinspace` in French (default) or independently in a French dialect using the optional argument. They are meant for LaTeX2e *only* and can only be used in the preamble. Four mandatory arguments are expected besides the optional one: the first one is a *string* either “guill”, “colon”, or “thin”, the last four are decimal numbers specifying *width*, *stretch* and *shrink* relative to *fontdimens*. For instance

`\FBsetspaces[acadian]{colon}{0.5}{0}{0}` defines `\acadianFBcolonspace` as a thinspace which will be used for the Acadian dialect only. When used without optional argument or with argument ‘french’, the same command would tune the basic `\FBcolonspace` command.

```

141 \ifLaTeXe
142 \newcommand*{\FBsetspaces}[5][french]{%
143 \def\bb1@tempa{french}\def\bb1@tempb{#1}%
144 \ifx\bb1@tempa\bb1@tempb \def\bb1@tempb{}\fi
145 \@namedef{\bb1@tempb FB#2space}{\hskip #3\fontdimen2\font
146 plus #4\fontdimen3\font
147 minus #5\fontdimen4\font \relax}%

```

With option “acadian”, fill the corresponding LuaTeX table. All unset values in the “acadian” subtables will be filled ‘AtBeginDocument’ by `\set@glue@table` with the value available for “french”.

```

148 \ifFB@luatex@punct
149 \ifx\bb1@tempb\FB@acadian
150 \directlua{
151 FBsp.#2.g1.ac[1] = #3
152 FBsp.#2.g1.ac[2] = #4
153 FBsp.#2.g1.ac[3] = #5
154 if #3 > 0.6 then
155 FBsp.#2.ch.ac = 0xA0
156 elseif #3 > 0.2 then
157 FBsp.#2.ch.ac = 0x202F
158 else
159 FBsp.#2.ch.ac = 0x200B
160 end
161 }%
162 \fi
163 \fi
164 }
165 \@onlypreamble\FBsetspaces
166 \fi

```

Remember that the *same* `\extrasfrench` command is executed when switching to French or to a French dialect (Acadian). Acadian and French may share the same patterns (or not), and may use different spacing for high punctuation and/or quotes. Basically, for pdfLaTeX and XeLaTeX, the spacing is set for French, then potentially tuned differently for Acadian. LuaTeX relies on an attribute `\FB@dialect` to decide what spacing is needed for French or Acadian (see LuaTeX table `FBsp`). As a rough test on `\languagename` would be unreliable to set the value of `\FB@dialect` (see `babel.pdf`), we use a trick based on `\detokenize`; another option would be to use the `\IfLanguageName` command from Oberdiek’s package `iflang`.

```

167 \ifLaTeXe
168 \addto\extrasfrench{%

```

```

169 \ifFB@luatex@punct
170 \edef\bbl@tempa{\detokenize\expandafter{\languagename}}}%
171 \edef\bbl@tempb{\detokenize{french}}}%
172 \ifx\bbl@tempa\bbl@tempb \FB@dialect=\z@
173 \else \FB@dialect=\@ne
174 \fi

```

When first entering French, we must set the LuaTeX tables for French (`\FB@dialect=0`) *before* any dialect redefines any `\FB...` space command. Doing this ‘AtBeginDocument’ would be too late: if French or a French dialect is the main language, `\extrasfrench` has been executed before!

```

175 \ifdefined\FB@once\else
176 \set@glue@table{colon}%
177 \set@glue@table{thin}%
178 \set@glue@table{guill}%
179 \def\FB@once{}%
180 \fi
181 \fi

```

Any dialect dependent customisation done using `\FBsetspaces[dialect]` command or alike is now taken into account: the value of `\FBthinspace` (meant for French, i.e. `\FB@dialect=0`) is first saved then changed (for Acadian).

```

182 \ifcsname\languagename FBthinspace\endcsname
183 \babel@save\FBthinspace
184 \renewcommand*{\FBthinspace}{%
185 \csname\languagename FBthinspace\endcsname}%
186 \fi

```

Same for `\FBcolonspace`:

```

187 \ifcsname\languagename FBcolonspace\endcsname
188 \babel@save\FBcolonspace
189 \renewcommand*{\FBcolonspace}{%
190 \csname\languagename FBcolonspace\endcsname}%
191 \fi

```

And for `\FBguillspace`:

```

192 \ifcsname\languagename FBguillspace\endcsname
193 \babel@save\FBguillspace
194 \renewcommand*{\FBguillspace}{%
195 \csname\languagename FBguillspace\endcsname}%
196 \fi
197 }
198 \fi

```

The conditional `\ifFB@spacing` will be used by pdfTeX and XeTeX engines to switch on or off space tuning before high punctuation and inside French quotes. A matching attribute will be defined later for LuaTeX.

```

199 \newif\ifFB@spacing \FB@spacingtrue

```

`\FB@spacing@off` Two internal commands to switch on and off all space tuning for all six characters `\FB@spacing@on` ‘;:!?«»’. They will be triggered by user command `\NoAutoSpacing` and by font family switching commands `\ttfamilyFB` `\rmfamilyFB` and `\sffamilyFB`. These four commands will now behave the same with any engine (up to version 3.2b, results were engine dependent).

```

200 \ifFB@luatex@punct
201   \newcommand*{\FB@spacing@on}{\FB@spacing=\@ne}
202   \newcommand*{\FB@spacing@off}{\FB@spacing=\z@}
203 \else
204   \newcommand*{\FB@spacing@on}{\FB@spacingtrue}
205   \newcommand*{\FB@spacing@off}{\FB@spacingfalse}
206 \fi

```

2.2.1 Punctuation with LuaTeX

The following part holds specific code for punctuation with modern LuaTeX engines, i.e. version 1.0.4 (included in TL2017) or newer.

```

207 \ifFB@luatex@punct
208   \ifdefined\newluafunction\else

```

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

```

209   \input ltxlua.tex
210 \fi

```

We define five LuaTeX attributes to control spacing in French and/or Acadian 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).

`\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@addGUILspace` will be set to 1 by option `og=«`, `fg=»`, thus enabling automatic insertion of proper spaces after ‘«’ and before ‘»’.

`\FB@ucsNBSP` triggers the replacement of glues by characters, it is controlled by option `UnicodeNoBreakSpaces`.

`\FB@dialect` is 0 for French and 1 for Acadian; its value controls which parts of the glue table (`.fr` or `.ac`) are taken into account.

```

211 \newattribute\FB@spacing      \FB@spacing=\@ne
212 \newattribute\FB@addDPspace  \FB@addDPspace=\@ne
213 \newattribute\FB@addGUILspace \FB@addGUILspace=\z@
214 \newattribute\FB@ucsNBSP     \FB@ucsNBSP=\z@
215 \newattribute\FB@dialect     \FB@dialect=\z@
216 \ifLaTeXe
217   \PackageInfo{french.ldf}{No need for active punctuation
218     characters\MessageBreak with this version

```

```

219             of LuaTeX!\MessageBreak reported}
220 \else
221   \fb@info{No need for active punctuation characters\\
222           with this version of LuaTeX!}
223 \fi

```

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 `\FBsetspace` commands) are taken into account. Values not explicitly set for Acadian by `\FBsetspace[acadian]` commands are copied from the French ones.

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.

```

224 \newcommand*{\set@glue@table}[1]{%
225   \directlua {
226     local s = token.get_meaning("FB#1space")
227     local t = FBget_glue(s)
228     if t then
229       FBsp.#1.gl.fr = t
230       if not FBsp.#1.gl.ac[1] then
231         FBsp.#1.gl.ac = t
232       end
233       if FBsp.#1.gl.fr[1] > 0.6 then
234         FBsp.#1.ch.fr = 0xA0
235       elseif FBsp.#1.gl.fr[1] > 0.2 then
236         FBsp.#1.ch.fr = 0x202F
237       else
238         FBsp.#1.ch.fr = 0x200B
239       end
240       if not FBsp.#1.ch.ac then
241         FBsp.#1.ch.ac = FBsp.#1.ch.fr
242       end
243     else
244       texio.write_nl('term and log', '')
245       texio.write_nl('term and log',
246         '*** french.ldf warning: Unexpected syntax in FB#1space,')
247       texio.write_nl('term and log',
248         '*** french.ldf warning: LuaTeX table FBsp unchanged.')
```



```

255 }
256 \fi
257 </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).

```

258 <*\lua>
259 local FB_punct_thin =
260   {[string.byte("!")] = true,
261    [string.byte("?")] = true,
262    [string.byte(";")] = true}
263 local FB_punct_thick =
264   {[string.byte(":")] = true}

```

Managing spacing after ‘«’ (U+00AB) and before ‘»’ (U+00BB) 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 ‘«’ 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 ‘»’.

```

265 local FB_punct_left =
266   {[string.byte("!")] = true,
267    [string.byte("?")] = true,
268    [string.byte(";")] = true,
269    [string.byte(":")] = true,
270    [0x14] = true,
271    [0xBB] = true}
272 local FB_punct_right =
273   {[0x13] = true,
274    [0xAB] = true}

```

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

```

275 local FB_punct_null =
276   {[string.byte("!")] = true,
277    [string.byte("?")] = true,
278    [string.byte("[")] = true,
279    [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.

```

280   [0xA0] = true,
281   [0x202F] = true}
282 local FB_guil_null =
283   {[0xA0] = true,
284    [0x202F] = true}

```

Local definitions for nodes:

```
285 local new_node      = node.new
286 local copy_node     = node.copy
287 local node_id       = node.id
288 local HLIST         = node_id("hlist")
289 local TEMP          = node_id("temp")
290 local DISC          = node_id("disc")
291 local KERN          = node_id("kern")
292 local GLUE          = node_id("glue")
293 local GLYPH         = node_id("glyph")
294 local PENALTY       = node_id("penalty")
295 local nobreak       = new_node(PENALTY)
296 nobreak.penalty    = 10000
297 local nbspace       = new_node(GLYPH)
298 local insert_node_before = node.insert_before
299 local insert_node_after  = node.insert_after
300 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.

```
301 function FBget_glue(toks)
302   local t = nil
303   local f = string.match(toks,
304     "[^%w]hskip%s*([%d%.]*)%s*[^%w]fontdimen 2")
305   if f == "" then f = 1 end
306   if tonumber(f) then
307     t = {tonumber(f), 0, 0}
308     f = string.match(toks, "plus%s*([%d%.]*)%s*[^%w]fontdimen 3")
309     if f == "" then f = 1 end
310     if tonumber(f) then
311       t[2] = tonumber(f)
312       f = string.match(toks, "minus%s*([%d%.]*)%s*[^%w]fontdimen 4")
313       if f == "" then f = 1 end
314       if tonumber(f) then
315         t[3] = tonumber(f)
316       end
317     end
318   elseif string.match(toks, "[^%w]F?B?thinspace") then
319     t = {0.5, 0, 0}
320   elseif string.match(toks, "[^%w]space") then
321     t = {1, 1, 1}
322   end
323   return t
324 end
```

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

```

325 FBsp = {}
326 FBsp.thin = {}
327 FBsp.thin.gl = {}
328 FBsp.thin.gl.fr = {.5, 0, 0} ; FBsp.thin.gl.ac = {}
329 FBsp.thin.ch = {}
330 FBsp.thin.ch.fr = 0x202F ; FBsp.thin.ch.ac = nil
331 FBsp.colon = {}
332 FBsp.colon.gl = {}
333 FBsp.colon.gl.fr = { 1, 1, 1} ; FBsp.colon.gl.ac = {}
334 FBsp.colon.ch = {}
335 FBsp.colon.ch.fr = 0xA0 ; FBsp.colon.ch.ac = nil
336 FBsp.guill = {}
337 FBsp.guill.gl = {}
338 FBsp.guill.gl.fr = {.8, .3, .8} ; FBsp.guill.gl.ac = {}
339 FBsp.guill.ch = {}
340 FBsp.guill.ch.fr = 0xA0 ; FBsp.guill.ch.ac = nil

```

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`.

```

341 local font_table = {}
342 local function new_glue_scaled (fid,table)
343   if fid > 0 and table[1] then
344     local fp = font_table[fid]
345     if not fp then
346       local ft = font.getfont(fid)
347       if ft then
348         font_table[fid] = ft.parameters
349         fp = font_table[fid]
350       end
351     end
352     local gl = new_node(GLUE,0)
353     if fp then
354       node.setglue(gl, table[1]*fp.space,
355                    table[2]*fp.space_stretch,
356                    table[3]*fp.space_shrink)
357       return gl
358     else
359       return nil
360     end
361   else
362     return nil

```

```

363 end
364 end

```

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

```

365 local FBspacing      = luatexbase.attributes['FB@spacing']
366 local addDPspace     = luatexbase.attributes['FB@addDPspace']
367 local addGUILspace   = luatexbase.attributes['FB@addGUILspace']
368 local FBucsNBSP     = luatexbase.attributes['FB@ucsNBSP']
369 local FBdialect     = luatexbase.attributes['FB@dialect']
370 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`). Constants `FR_fr` (french) and `FR_ca` (acadian) are defined by command `\activate@luatexpunct`.

```

371 -- Main function (to be added to the kerning callback).
372 local function french_punctuation (head)

```

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

```

373   node.kerning(head)
374   for item in node.traverse_id(GLYPH, head) do
375     local lang = item.lang
376     local char = item.char

```

Skip glyphs not concerned by French kernings.

```

377     if (lang == FR_fr or lang == FR_ca) and
378         (FB_punct_left[char] or FB_punct_right[char]) then
379       local fid = item.font
380       local attr = item.attr
381       local FRspacing = has_attribute(item, FBspacing)
382       FRspacing = FRspacing and FRspacing > 0
383       local FRucsNBSP = has_attribute(item, FBucsNBSP)
384       FRucsNBSP = FRucsNBSP and FRucsNBSP > 0
385       local FRdialect = has_attribute(item, FBdialect)
386       FRdialect = FRdialect and FRdialect > 0
387       local SIG = has_attribute(item, addGUILspace)
388       SIG = SIG and SIG > 0
389       if FRspacing and fid > 0 then
390         if FB_punct_left[char] then
391           local prev = item.prev
392           local prev_id, prev_subtype, prev_char
393           if prev then
394             prev_id = prev.id

```

```

395         prev_subtype = prev.subtype
396         if prev_id == GLYPH then
397             prev_char = prev.char
398         end
399     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.

```

400         local is_glue = prev_id == GLUE
401         local glue_wd
402         if is_glue then
403             glue_wd = prev.width
404         end
405         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.

```

406         if FB_punct_thin[char] or FB_punct_thick[char] then
407             local SBDP = has_attribute(item, addDPspace)
408             local auto = SBDP and SBDP > 0
409             if FB_punct_thick[char] and auto then
410                 local next = item.next
411                 local next_id
412                 if next then
413                     next_id = next.id
414                 end
415                 if next_id and
416                     (next_id == GLYPH or next_id == DISC) then
417                     auto = false
418                 end
419             end
420             if auto then
421                 if (prev_char and FB_punct_null[prev_char]) or
422                     (is_glue and glue_wd ≤ 1) or
423                     (prev_id == HLIST and prev_subtype == 3) or

```

```

424         (prev_id == TEMP) then
425         auto = false
426         end
427     end
428     local fbglue
429     local t
430     if FB_punct_thick[char] then
431         if FRdialect then
432             t = FBsp.colon.gl.ac
433             nbspace.char = FBsp.colon.ch.ac
434         else
435             t = FBsp.colon.gl.fr
436             nbspace.char = FBsp.colon.ch.fr
437         end
438     else
439         if FRdialect then
440             t = FBsp.thin.gl.ac
441             nbspace.char = FBsp.thin.ch.ac
442         else
443             t = FBsp.thin.gl.fr
444             nbspace.char = FBsp.thin.ch.fr
445         end
446     end
447     fbglue = new_glue_scaled(fid, t)

```

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

```

448     if (realglue or auto) and fbglue then
449         if realglue then
450             head = remove_node(head,prev,true)
451         end
452         if (FRucsNBSP) then
453             nbspace.font = fid
454             nbspace.attr = attr
455             insert_node_before(head,item,copy_node(nbspace))
456         else
457             nobreak.attr = attr
458             fbglue.attr = attr
459             insert_node_before(head,item,copy_node(nobreak))
460             insert_node_before(head,item,copy_node(fbglue))
461         end
462     end

```

Let's consider '»' now (the only remaining glyph of `FB_punct_left` class): we just have to remove any *glue* possibly preceding '»', then to insert the nobreak penalty and the proper *glue* (controlled by `\FBguillspace`). This is done only if French quotes have been 'activated' by options `og=«`, `fg=»` in `\frenchsetup{}` and can be denied locally with `\NoAutoSpacing` (this is controlled by the SIG flag). If either a) the pre-

ceding 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.

```

463         elseif SIG then
464             local addgl = (prev_char and
465                 not FB_guil_null[prev_char])
466                 or
467                 (not prev_char and
468                 prev_id ~= TEMP and
469                 not (prev_id = HLIST and
470                     prev_subtype = 3)
471             )

```

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

```

472         if is_glue and glue_wd ≤ 1 then
473             addgl = false
474         end
475         local t = FBsp.guill.gl.fr
476         nbspace.char = FBsp.guill.ch.fr
477         if FRdialect then
478             t = FBsp.guill.gl.ac
479             nbspace.char = FBsp.guill.ch.ac
480         end
481         local fbg glue = new_glue_scaled(fid, t)
482         if addgl and fbg glue then
483             if is_glue then
484                 head = remove_node(head,prev,true)
485             end
486             if (FRucsNBSP) then
487                 nbspace.font = fid
488                 nbspace.attr = attr
489                 insert_node_before(head,item,copy_node(nbspace))
490             else
491                 nobreak.attr = attr
492                 fbg glue.attr = attr
493                 insert_node_before(head,item,copy_node(nobreak))
494                 insert_node_before(head,item,copy_node(fbg glue))
495             end
496         end
497     end

```

Similarly, for '«' (unique member 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*.

```

498         elseif SIG then
499             local next = item.next

```



```

500         local next_id, next_subtype, next_char, nextnext, kern_wd
501         if next then
502             next_id = next.id
503             next_subtype = next.subtype

```

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

```

504             if next_id == PENALTY then
505                 nextnext = next.next
506                 if nextnext and nextnext.id == GLUE then
507                     head = remove_node(head,nextnext,true)
508                     head = remove_node(head,next,true)
509                     next = item.next
510                     if next then
511                         next_id = next.id
512                         next_subtype = next.subtype
513                         if next_id == GLYPH then
514                             next_char = next.char
515                         end
516                     end
517                 end
518             end

```

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

```

519             if next_id == KERN then
520                 kern_wd = next.kern
521                 if kern_wd ==  $\emptyset$  then
522                     nextnext = next.next
523                     if nextnext then
524                         next = nextnext
525                         next_id = nextnext.id
526                         next_subtype = nextnext.subtype
527                         if next_id == PENALTY then
528                             nextnext = next.next
529                             if nextnext and nextnext.id == GLUE then
530                                 head = remove_node(head,next,true)
531                                 head = remove_node(head,nextnext,true)
532                                 next = item.next
533                                 if next then
534                                     next_id = next.id
535                                     next_subtype = next.subtype
536                                 end
537                             end
538                         end
539                     end
540                 end
541             end

```

```

542         if next_id == GLYPH then
543             next_char = next.char
544         end
545     end
546     local is_glue = next_id == GLUE
547     if is_glue then
548         glue_wd = next.width
549     end

```

The `addgl` flag only depends on `next_char` and `is_glue`:

```

550         local addgl = (next_char and not FB_guil_null[next_char])
551                     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).

```

552         if is_glue and glue_wd == 0 then
553             addgl = false
554         end
555         local fid = item.font
556         local t = FBsp.guill.gl.fr
557         nbspace.char = FBsp.guill.ch.fr
558         if FRdialect then
559             t = FBsp.guill.gl.ac
560             nbspace.char = FBsp.guill.ch.ac
561         end
562         local fbglue = new_glue_scaled(fid, t)
563         if addgl and fbglue then
564             if is_glue then
565                 head = remove_node(head,next,true)
566             end
567             if (FRucsNBSP) then
568                 nbspace.font = fid
569                 nbspace.attr = attr
570                 insert_node_after(head, item, copy_node(nbspace))
571             else
572                 nobreak.attr = attr
573                 fbglue.attr = attr
574                 insert_node_after(head, item, copy_node(fbglue))
575                 insert_node_after(head, item, copy_node(nobreak))
576             end
577         end
578     end
579 end
580 end
581 end
582 return head
583 end
584 return french_punctuation

```

585 </lua>

As a language tag is part of glyph nodes in LuaTeX, no more switching has to be done in `\extrasfrench`, setting the dialect attribute has already been done (see above, p. 19). 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`).

```
586 <{*french>
587 \ifFB@luatex@punct
588   \def\activate@luatexpunct{%
589     \directlua{%
590       FR_fr = \the\l@french ; FR_ca = \the\l@acadian ;
591       local path = kpse.find_file("frenchb.lua", "lua")
592       if path then
593         local f = dofile(path)
594         luatexbase.add_to_callback("kerning",
595           f, "frenchb.french_punctuation")
596       else
597         texio.write_nl('')
598         texio.write_nl('*****')
599         texio.write_nl('Error: frenchb.lua not found.')
600         texio.write_nl('*****')
601         texio.write_nl('')
602       end
603     }%
604   }
605 \fi
```

End of specific code for punctuation with LuaTeX engines.

2.2.2 Punctuation with XeTeX

If `\XeTeXinterchartokenstate` is available, we use the “inter char” mechanism to provide correct spacing in French before the four characters ; ! ? and :. The basis of the following code was borrowed from the `polyglossia` package, see `gloss-french.ldf`. We use the same mechanism for French quotes (« and »), when automatic spacing for quotes is required by options `og=«` and `fg=»` in `\frenchsetup{}` (see section 2.11). Unless `ucharclass` is loaded, the default value for `\XeTeXcharclass` is 0 for characters tokens and `\FB@nonchar` for all other tokens (glues, kerns, math and box boundaries, etc.). `ucharclass` defines a XeTeX class for every range of Unicode characters in order to facilitate font switching. Most French characters belong to range [”20, ”7F] (class `\BasicLatinClass`) some (accented chars, diacritics,...) to range [”80, ”FF] (class `\LatinSupplementClass`) and three (œ, Œ, and the long-s) to [”100, ”17F] (class `\LatinExtendedAClass`).

We check `AtBeginDocument` whether `ucharclass` is loaded; if so, when switching to French, the class `\FB@stdchar` of all characters possibly used in French (except punctuation) will be forced to `\BasicLatinClass` which is the default for most of them, the class of the others (accented chars, ligatures, diacritics, etc.) will be saved and changed locally in French, then restored to their original value when leaving French.

We switch `\XeTeXinterchartokenstate` to 1 and change the `\XeTeXcharclass` values of `! ? : (] «` and `»` when entering French. Their initial values will be restored when leaving French.

The following part holds specific code for punctuation with XeTeX engines.

```

606 \ifFB@xetex@punct
607   \ifLaTeXe
608     \PackageInfo{french.ldf}{No need for active punctuation
609                           characters\MessageBreak with this
610                           version of XeTeX!\MessageBreak reported}
611   \else
612     \fb@info{No need for active punctuation characters\
613             with this version of XeTeX!}
614   \fi

```

Six new character classes are defined for `babel-french`.

```

615 \newXeTeXintercharclass\FB@punctthick
616 \newXeTeXintercharclass\FB@punctthin
617 \newXeTeXintercharclass\FB@punctnul
618 \newXeTeXintercharclass\FB@guilo
619 \newXeTeXintercharclass\FB@guilf
620 \newXeTeXintercharclass\FB@guilnul

```

As `\babel@savevariable` doesn't work inside a `\bb1@for` loop, we define a variant to save the `\XeTeXcharclass` values which will be modified in French.

```

621 \def\FBsavevariable@loop#1#2{\begingroup
622   \toks@\expandafter{\originalTeX #1}%
623   \edef\x{\endgroup
624     \def\noexpand\originalTeX{\the\toks@ #2=\the#1#2\relax}}%
625   \x}

```

`\FB@charlistsave` holds the all list of characters which have their `\XeTeXcharclass` value modified in French: it always includes high punctuation, French quotes, opening delimiters and no-break spaces. If `ucharclasses` is loaded, non-ASCII characters used in French have to be added; as `xeCJK` changes the class of some characters used in French, these have to be saved too if `xeCJK` is loaded.

```

626 \def\FB@charlist{"21,"3A,"3B,"3F,"AB,"BB,"28,"5B,"A0,"202F}
627 \def\FB@charlistUCC{}
628 \def\FB@charlistxeCJK{}
629 \edef\FB@charlistsave{\FB@charlist}
630 \ifLaTeXe

```

```

631 \AtBeginDocument{%
632   \@ifpackageloaded{ucharclasses}%
633   {\ifdefined\BasicLatinClass
634     \RenewCommandCopy{\FB@stdchar}{\BasicLatinClass}%
635     \def\FB@charlistUCC{"C0,"C2,"C6,"C7,"C8,"C9,"CA,"CB,"CE,"CF,%
636       "D4,"D6,"D9,"DB,"DC,"E0,"E2,"E6,"E7,"E8,"E9,"EA,"EB,"EE,%
637       "EF,"F4,"F6,"F9,"FB,"FC,"152,"153,"17F,"2019}%
638     \addto\FB@charlist{\FB@charlistUCC}%
639     \edef\FB@charlistsave{\FB@charlist}%
640     \fi
641   }%
642   \@ifpackageloaded{xeCJK}%
643   {\def\FB@charlistxeCJK%
644     "29,"5D,"7B,"7D,"2C,"2D,"2E,"22,"25,"27,"60,"2019}%
645     \addto\FB@charlist{\FB@charlistxeCJK}%
646     \edef\FB@charlistsave{\FB@charlist}%
647     }%
648   }
649 \fi

```

`\FB@xetex@punct@french` The following command will be executed when entering French, it first saves the values to be modified, then fits them to our needs.

```

650 \newcommand*{\FB@xetex@punct@french}{%
651   \babel@savevariable{\XeTeXinterchartokenstate}%
652   \bbl@for\FB@char\FB@charlistsave
653     {\FBsavevariable@loop{\XeTeXcharclass}{\FB@char}}%

```

If `ucharclasses` is loaded, force non-ASCII used in French to class `\FB@stdchar` (`=\BasicLatinClass`).

```

654 \ifx\FB@charlistUCC\@empty\else
655   \bbl@for\FB@char\FB@charlistUCC
656     {\XeTeXcharclass \FB@char \FB@stdchar}%
657 \fi

```

These characters have their class changed by `xeCJK.sty`, let's reset their class in French.

```

658 \ifx\FB@charlistxeCJK\@empty\else
659   \bbl@for\FB@char\FB@charlistxeCJK
660     {\XeTeXcharclass\FB@char=\FB@stdchar}%
661 \fi

```

This will avoid spurious spaces in (!), [?] and with Unicode non-breaking spaces (U+00A0, U+202F):

```

662 \bbl@for\FB@char {\`[,`\[,`\"A0,\"202F}%
663   {\XeTeXcharclass\FB@char=\FB@punctnul}%

```

Let's now define specific classes for punctuation and interactions between classes. When false, the flag `\ifFB@spacing` switches off any interaction between classes (this

flag is controlled by user-level command `\NoAutoSpacing`; this flag is also set to false when the current font is a typewriter font).

```

664 \XeTeXinterchartokenstate=\@ne
665 \XeTeXcharclass `\: = \FB@punctthick
666 \XeTeXinterchartoks \FB@stdchar \FB@punctthick = {%
667   \ifFB@spacing\ifhmode\FDP@colonspace\fi\fi}%
668 \XeTeXinterchartoks \FB@guilf \FB@punctthick = {%
669   \ifFB@spacing\FDP@colonspace\fi}%

```

Small glues such as “glue 1sp” in tabular ‘l’ columns or “glue 0 plus 1 fil” in tabular ‘c’ columns or `\lstlisting` environment should not trigger any extra space; they will still do when `AutoSpacePunctuation` is true: `\XeTeXcharclass=\FB@nonchar` isn’t specific to glue tokens (this class includes box and math boundaries f.i.), so the `\else` part cannot be omitted.

```

670 \XeTeXinterchartoks \FB@nonchar \FB@punctthick = {%
671   \ifFB@spacing
672     \ifhmode
673       \ifdim\lastskip>1sp
674         \unskip\penalty\@M\FBcolonspace
675       \else
676         \FDP@colonspace
677       \fi
678     \fi
679   \fi}%
680 \bbl@for\FB@char {\`;\,`!\,`?}%
681   {\XeTeXcharclass\FB@char=\FB@punctthin}%
682 \XeTeXinterchartoks \FB@stdchar \FB@punctthin = {%
683   \ifFB@spacing\ifhmode\FDP@thinspace\fi\fi}%
684 \XeTeXinterchartoks \FB@guilf \FB@punctthin = {%
685   \ifFB@spacing\FDP@thinspace\fi}%
686 \XeTeXinterchartoks \FB@nonchar \FB@punctthin = {%
687   \ifFB@spacing
688     \ifhmode
689       \ifdim\lastskip>1sp
690         \unskip\penalty\@M\FBthinspace
691       \else
692         \FDP@thinspace
693       \fi
694     \fi
695   \fi}%
696 \XeTeXinterchartoks \FB@guilo \FB@stdchar = {%
697   \ifFB@spacing\FB@guillspace\fi}%
698 \XeTeXinterchartoks \FB@guilo \FB@nonchar = {%
699   \ifFB@spacing\FB@guillspace\ignorespaces\fi}%
700 \XeTeXinterchartoks \FB@stdchar \FB@guilf = {%
701   \ifFB@spacing\FB@guillspace\fi}%

```

```

702 \XeTeXinterchartoks \FB@punctthin \FB@guilf = {%
703   \ifFB@spacing\FB@guillspace\fi}%
704 \XeTeXinterchartoks \FB@nonchar \FB@guilf = {%
705   \ifFB@spacing\unskip\FB@guillspace\fi}%
706 }
707 \addto\extrasfrench{\FB@xetex@punct@french}

```

End of specific code for punctuation with modern XeTeX engines.

```
708 \fi
```

2.2.3 Punctuation with standard (pdf)TeX

In standard (pdf)TeX we need to make the four characters ; ! ? and : ‘active’ and provide their definitions. Before doing so, we have to save some definitions involving `..`:

```

709 \newif\ifFB@koma
710 \ifLaTeXe
711   \@ifclassloaded{scrartcl}{\FB@komatrue}{}
712   \@ifclassloaded{scrbook}{\FB@komatrue}{}
713   \@ifclassloaded{scrreprt}{\FB@komatrue}{}
714   \ifFB@koma\def\FB@std@capsep{: \ } \fi
715   \@ifclassloaded{beamer}{\def\FB@std@capsep{: }}{}
716   \@ifclassloaded{memoir}{\def\FB@std@capsep{: }}{}
717 \fi

718 \ifFB@active@punct
719   \initiate@active@char{:}%
720   \initiate@active@char{;}%
721   \initiate@active@char{!}%
722   \initiate@active@char{?}%

```

We first tune the amount of space before ; ! ? and :. This should only happen in horizontal mode, hence the test `\ifhmode`.

In horizontal mode, if a space has been typed before ‘;’ we remove it and put a non-breaking `\FBthinspace` instead. If no space has been typed, we add `\FDP@thinspace` which will be defined, up to the user’s wishes, as a non-breaking `\FBthinspace` or as `\@empty`.

```

723 \declare@shorthand{french}{};{}%
724 \ifFB@spacing
725   \ifhmode
726     \ifdim\lastskip>1sp
727       \unskip\penalty\@M\FBthinspace
728     \else
729       \FDP@thinspace
730     \fi
731   \fi
732 \fi

```

Now we can insert a ; character.

```
733   \string;}
```

The next three definitions are very similar.

```
734 \declare@shorthand{french}{!}{}%  
735   \ifFB@spacing  
736     \ifhmode  
737       \ifdim\lastskip>1sp  
738         \unskip\penalty\@M\FBthinspace  
739       \else  
740         \FDP@thinspace  
741     \fi  
742   \fi  
743 \fi  
744 \string!}  
745 \declare@shorthand{french}{?}{}%  
746   \ifFB@spacing  
747     \ifhmode  
748       \ifdim\lastskip>1sp  
749         \unskip\penalty\@M\FBthinspace  
750       \else  
751         \FDP@thinspace  
752     \fi  
753   \fi  
754 \fi  
755 \string?}  
756 \declare@shorthand{french}{:}{}%  
757   \ifFB@spacing  
758     \ifhmode  
759       \ifdim\lastskip>1sp  
760         \unskip\penalty\@M\FBcolonspace  
761       \else  
762         \FDP@colonspace  
763     \fi  
764   \fi  
765 \fi  
766 \string:}
```

When the active characters appear in an environment where their French behaviour is not wanted they should give an ‘expected’ result. Therefore we define shorthands at system level as well.

```
767 \declare@shorthand{system}{:}{\string:}  
768 \declare@shorthand{system}{!}{\string!}  
769 \declare@shorthand{system}{?}{\string?}  
770 \declare@shorthand{system}{;}{\string;}
```

We specify that the French group of shorthands should be used when switching to French.


```
771 \addto\extrasfrench{\languageshorthands{french}}%
```

These characters are ‘turned on’ once, later their definition may vary. Don’t misunderstand the following code: they keep being active all along the document, even when leaving French.

```
772 \bbl@activate{:}\bbl@activate{;}%
773 \bbl@activate{!}\bbl@activate{?}%
774 }
775 \addto\noextrasfrench{%
776 \bbl@deactivate{:}\bbl@deactivate{;}%
777 \bbl@deactivate{!}\bbl@deactivate{?}%
778 }
779 \fi
```

2.2.4 Punctuation switches common to all engines

A new ‘if’ `\iffBFAutoSpacePunctuation` 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.

```
780 \newif\iffBFAutoSpacePunctuation \FBAutoSpacePunctuationtrue
```

`\AutoSpaceBeforeFDP` `\autospace@beforeFDP` and `\noautospace@beforeFDP` are internal commands. `\NoAutoSpaceBeforeFDP` `\autospace@beforeFDP` defines commands `\FDP@thinspace` and `\FDP@colonspace` as non-breaking spaces and sets LuaTeX attribute `\FB@addDPspace` to 1 (true), while `\noautospace@beforeFDP` makes them no-op and sets flag `\FB@addDPspace` to 0 (false). User commands `\AutoSpaceBeforeFDP` and `\NoAutoSpaceBeforeFDP` do the same and take care of the flag `\iffBFAutoSpacePunctuation` in LaTeX. Set the default now for Plain (done later for LaTeX).

```
781 \def\autospace@beforeFDP{%
782 \iffB@luatex@punct \FB@addDPspace=\@ne \fi
783 \def\FDP@thinspace{\penalty\@M\FBthinspace}%
784 \def\FDP@colonspace{\penalty\@M\FBcolonspace}}
785 \def\noautospace@beforeFDP{%
786 \iffB@luatex@punct \FB@addDPspace=\z@ \fi
787 \let\FDP@thinspace\@empty
788 \let\FDP@colonspace\@empty}
789 \ifLaTeXe
790 \def\AutoSpaceBeforeFDP{\autospace@beforeFDP
791 \FBAutoSpacePunctuationtrue}
792 \def\NoAutoSpaceBeforeFDP{\noautospace@beforeFDP
793 \FBAutoSpacePunctuationfalse}
794 \AtEndOfPackage{\AutoSpaceBeforeFDP}
795 \else
796 \let\AutoSpaceBeforeFDP\autospace@beforeFDP
797 \let\NoAutoSpaceBeforeFDP\noautospace@beforeFDP
798 \AutoSpaceBeforeFDP
```

799 `\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 *when they are typed in as characters* with the ‘og’/‘fg’ options in `\frenchsetup{}`. This is also a workaround for the weird behaviour of these characters in verbatim mode.

```
800 \ifLaTeXe
801 \DeclareRobustCommand\ttfamilyFB{\FB@spacing@off \ttfamilyORI}
802 \DeclareRobustCommand\rmfamilyFB{\FB@spacing@on \rmfamilyORI}
803 \DeclareRobustCommand\sffamilyFB{\FB@spacing@on \sffamilyORI}
804 \fi
```

`\NoAutoSpacing` The following command disables automatic spacing for high punctuation and French quote characters; it also switches off active punctuation characters (if any). It is engine independent (works for TeX, LuaTeX and XeTeX based engines) and is meant to be used inside a group. The faked definition of `\texorpdfstring` will be overwritten by `hyperref.sty`.

```
805 \providecommand\texorpdfstring[2]{#1}
806 \DeclareRobustCommand*\NoAutoSpacing{}%
807 \texorpdfstring{\FB@spacing@off
808 \ifFB@active@punct\shorthandoff{;:!?}\fi}{}%
809 }
```

2.3 Commands for French quotation marks

`\guillemotleft` pdfLaTeX users are supposed to use 8-bit output encodings (T1, LY1,...) to type-

`\guillemotright` set French, those who still stick to OT1 should load `aeguill` or a similar package.

`\textquotedblleft` In both cases the commands `\guillemotleft` and `\guillemotright` will print the

`\textquotedblright` French opening and closing quote characters from the output font. For XeLaTeX and LuaLaTeX, `\guillemotleft` and `\guillemotright` are defined by package `fontspec` (v. 2.5d and up).

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

```
810 \ifLaTeXe
```

```

811 \else
812   \ifFBunicode
813     \def\guillemotleft{\char"00AB}
814     \def\guillemotright{\char"00BB}
815     \def\textquotedblleft{\char"201C}
816     \def\textquotedblright{\char"201D}
817   \else
818     \def\guillemotleft{\leavevmode\raise0.25ex
819       \hbox{\scriptscriptstyle\l1}}
820     \def\guillemotright{\raise0.25ex
821       \hbox{\scriptscriptstyle\gg}}
822     \def\textquotedblleft{``}
823     \def\textquotedblright{' '}
824   \fi
825   \let\xspace\relax
826 \fi

```

`\FBgspchar` The next step is to provide correct spacing after ‘<’ and before ‘>’; no line break is allowed neither *after* the opening one, nor *before* the closing one. French quotes `\FB@og` (including spacing) are printed by `\FB@og` and `\FB@fg`, the expansion of the top level commands `\og` and `\fg` is different in and outside French.

`\FB@og` and `\FB@fg` are now designed to work in bookmarks.

```

827 \newcommand*{\FB@og}{\texorpdfstring{\@FB@og}{\guillemotleft\space}}
828 \newcommand*{\FB@fg}{\texorpdfstring{\@FB@fg}{\space\guillemotright}}

```

The internal definitions `\@FB@og` and `\@FB@fg` need some engine-dependent tuning: for LuaTeX, `\FB@spacing` is set to 0 locally to prevent the quotes characters from adding space when option `og=«, fg=»` is set.

```

829 \newcommand*{\FB@guillspace}{\penalty\@M\FBguillspace}
830 \newcommand*{\FBgspchar}{\char"A0\relax}
831 \newif\ifFBucsNBSP
832 \ifFB@luatex@punct
833   \DeclareRobustCommand*{\@FB@og}{\leavevmode
834     \bgroup\FB@spacing=\z@ \guillemotleft\egroup
835     \ifFBucsNBSP\FBgspchar\else\FB@guillspace\fi}
836   \DeclareRobustCommand*{\@FB@fg}{\ifdim\lastskip>\z@\unskip\fi
837     \ifFBucsNBSP\FBgspchar\else\FB@guillspace\fi
838     \bgroup\FB@spacing=\z@ \guillemotright\egroup}
839 \fi

```

With XeTeX, `\ifFB@spacing` is set to false locally for the same reason.

```

840 \ifFB@xetex@punct
841   \DeclareRobustCommand*{\@FB@og}{\leavevmode
842     \bgroup\FB@spacingfalse\guillemotleft\egroup
843     \FB@guillspace}
844   \DeclareRobustCommand*{\@FB@fg}{\ifdim\lastskip>\z@\unskip\fi

```

```

845     \FB@guilospace
846     \bgroup\FB@spacingfalse\guillemotright\egroup}
847 \fi
848 \ifFB@active@punct
849   \DeclareRobustCommand*{\@FB@og}{\leavevmode
850     \guillemotleft
851     \FB@guilospace}
852   \DeclareRobustCommand*{\@FB@fg}{\ifdim\lastskip>\z@\unskip\fi
853     \FB@guilospace
854     \guillemotright}
855 \fi

```

`\og` The user level macros for quotation marks are named `\og` (“ouvrez guillemets”) and `\fg` (“fermez guillemets”). Another option for typesetting quotes in French is to use the command `\frquote` (see below). Dummy definition of `\og` and `\fg` just to ensure that this commands are not yet defined.

```

856 \newcommand*{\og}{\@empty}
857 \newcommand*{\fg}{\@empty}

```

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).

```

858 \ifLaTeXe
859   \def\bb1@frenchguillemets{%
860     \renewcommand*{\og}{\FB@og}%
861     \renewcommand*{\fg}{\FB@fg\xspace}}
862   \renewcommand*{\og}{\textquotedblleft}
863   \renewcommand*{\fg}{\ifdim\lastskip>\z@\unskip\fi
864     \textquotedblright\xspace}
865 \else
866   \def\bb1@frenchguillemets{\let\og\FB@og
867     \let\fg\FB@fg}
868   \def\og{\textquotedblleft}
869   \def\fg{\ifdim\lastskip>\z@\unskip\fi\textquotedblright}
870 \fi

871 \addto\extrasfrench{\babel@save\og \babel@save\fg
872   \bb1@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...

```

873 \newcommand*{\ogi}{\FB@og}

```

```

874 \newcommand*{\fgi}{\FB@fg}
875 \newcommand*{\@ogi}{\ifmmode\hbox{\ogi}\else\ogi\fi}
876 \newcommand*{\@fgi}{\ifmmode\hbox{\fgi}\else\fgi\fi}
877 \newcommand*{\ogii}{\textquotedblleft}
878 \newcommand*{\fgii}{\textquotedblright}
879 \newcommand*{\@ogii}{\ifmmode\hbox{\ogii}\else\ogii\fi}
880 \newcommand*{\@fgii}{\ifmmode\hbox{\fgii}\else\fgii\fi}

```

and the needed technical stuff to handle options:

```

881 \newcount\FBguill@level
882 \newtoks\FBold@everypar

```

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

```

883 \def\FB@addquote@everypar{%
884   \let\FBnew@everypar\everypar
885   \FBold@everypar=\expandafter{\the\everypar}%
886   \FBnew@everypar={\the\FBold@everypar\FBeverypar@quote}%
887   \let\everypar\FBold@everypar
888   \let\FB@addquote@everypar\relax
889 }
890 \newif\ifFBcloseguill \FBcloseguilltrue
891 \newif\ifFBInnerGuillSingle
892 \def\FBguillopen{\bgroup\NoAutoSpacing\guillemotleft\egroup}
893 \def\FBguillclose{\bgroup\NoAutoSpacing\guillemotright\egroup}
894 \let\FBguillnone\empty
895 \let\FBeveryparguill\FBguillopen
896 \let\FBverylineguill\FBguillnone
897 \let\FBeverypar@quote\relax
898 \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.

```

899 \ifLaTeXe
900   \DeclareRobustCommand\frquote{%
901     \texorpdfstring{\@ifstar{\FBcloseguillfalse\fr@quote}%
902                       {\FBcloseguilltrue \fr@quote}}%
903     {\bm@fr@quote}%
904   }
905   \newcommand{\bm@fr@quote}[1]{%
906     \guillemotleft\space #1\space\guillemotright}
907 \else
908   \newcommand\frquote[1]{\fr@quote{#1}}
909 \fi

```

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

```

910 \newcommand{\fr@quote}[1]{%

```

```

911 \leavevmode
912 \advance\FBguill@level by \@ne
913 \ifcase\FBguill@level
914 \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:

```

915 \ifx\FBeveryparguill\FBguillnone
916 \else
917 \def\FBeverypar@quote{\FBeveryparguill\FB@guillspace}%
918 \FB@addquote@everypar
919 \fi
920 \@ogi #1\@fgi
921 \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.

```

922 \ifx\FBverylineguill\FBguillopen
923 \def\FBveryline@quote{\FB@addGUllspace=\z@
924 \guillemotleft\FBguillspace}%
925 \localleftbox{\FBveryline@quote}%
926 \let\FBeverypar@quote\relax
927 \@ogi #1\ifFBcloseguill\@fgi\fi
928 \else
929 \ifx\FBverylineguill\FBguillclose
930 \def\FBveryline@quote{\FB@addGUllspace=\z@
931 \guillemotright\FBguillspace}%
932 \localleftbox{\FBveryline@quote}%
933 \let\FBeverypar@quote\relax
934 \@ogi #1\ifFBcloseguill\@fgi\fi
935 \else

```

otherwise we need to redefine `\FBeverypar@quote` (and eventually `\ogii`, `\fgii`) for level 2 quotations:

```

936 \let\FBeverypar@quote\relax
937 \ifFBInnerGuillSingle
938 \def\ogii{\leavevmode
939 \guilsinglleft\FB@guillspace}%
940 \def\fgii{\ifdim\lastskip>\z@\unskip\fi
941 \FB@guillspace\guilsinglright}%
942 \ifx\FBeveryparguill\FBguillopen
943 \def\FBeverypar@quote{\guilsinglleft\FB@guillspace}%
944 \fi
945 \ifx\FBeveryparguill\FBguillclose
946 \def\FBeverypar@quote{\guilsinglright\FB@guillspace}%
947 \fi
948 \fi
949 \@ogii #1\ifFBcloseguill \@fgii \fi

```

```

950     \fi
951     \fi
952   \else
Warn if \FBguill@level > 2:
953     \ifx\PackageWarning\@undefined
954       \fb@warning{\noexpand\frquote\space handles up to
955                 two levels.\\ Quotation not printed.}%
956     \else
957       \PackageWarning{french.1df}{%
958         \protect\frquote\space handles up to two levels.
959         \MessageBreak Quotation not printed. Reported}
960     \fi
961   \fi

```

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

```

962   \global\advance\FBguill@level by \m@ne
963   \ifcase\FBguill@level \global\let\FBeverypar@quote\relax
964   \or \gdef\FBeverypar@quote{\FBeveryparguill\FB@guillspace}%
965     \global\let\FBeverylin@quote\empty
966     \ifx\FBeverylin@guill\FBguillnone\else\localleftbox{}\fi
967   \fi
968 }

```

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.

```

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

```

2.4 Date in French

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

Explicitly defining `\BabelLanguages` as the list of all French dialects defines *both* `\datefrench` and `\dateacadian`; this is required as `french.1df` is read only once even if both language options `french` and `acadian` are supplied to Babel. Coding `\StartBabelCommands*{french,acadian}` would *only* define `\date\CurrentOption`, leaving the second language undefined in Babel's sens.

```

970 \def\BabelLanguages{french,acadian}
971 \StartBabelCommands*\BabelLanguages}{date}
972   [unicode, fontenc=TU EU1 EU2, charset=utf8]
973   \SetString\monthiiname{février}
974   \SetString\monthviiiname{août}

```

```

975 \SetString\monthxiiname{décembre}
976 \StartBabelCommands*\BabelLanguages\{date}
977 \SetStringLoop\month#1name\{%
978     janvier,f\evrier,mars,avril,mai,juin,juillet,%
979     ao\^ut,septembre,octobre,novembre,d\ecembre}
980 \SetString\today{\FB@date\year\month\day}
981 \EndBabelCommands

```

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

```

982 \newcommand*\FB@date[3]{%
983     {\number#3}\ifnum1=#3\ier\fi\FBdatespace
984     \csname month\romannumeral#2name\endcsname
985     \ifx#1\@empty\else\FBdatespace\number#1\fi}
986 \newcommand*\FBdatebox\hbox
987 \newcommand*\FBdatespace\space
988 \newcommand*\frenchdate\FBdatebox\FB@date
989 \newcommand*\acadiandate\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}'. Up to version 2.0 of `babel-french` `\up` was just a shortcut for `\textsuperscript` in LaTeX2e, but several users complained that `\textsuperscript` typesets superscripts too high and too big, so we now define `\fup` as an attempt to produce better looking superscripts. `\up` is defined as `\fup` but `\frenchsetup{FrenchSuperscripts=false}` redefines `\up` as `\textsuperscript` for compatibility with previous versions.

When a font has built-in superscripts, the best thing to do is to just use them, otherwise `\fup` has to simulate superscripts by scaling and raising ordinary letters. 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).

```

990 \newif\ifFB@poorman
991 \newdimen\FB@Mht
992 \ifLaTeXe
993 \AtEndOfPackage{\RequirePackage{scalegnt}}

```

`\FB@up@fake` 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. The chosen settings look correct for most fonts, but can be tuned by the end-user if necessary by changing `\FBsupR` and `\FBsupS` commands.

`\FB@1c` is defined as `\MakeLowercase` to inhibit the uppercasing of superscripts (this may happen in page headers with the standard classes but is wrong); `\FB@1c` can be redefined to do nothing by option `LowercaseSuperscripts=false` of `\frenchsetup`.


```

994 \newcommand*\FBsupR}{-0.12}
995 \newcommand*\FBsupS}{0.65}
996 \newcommand*\FB@lc}[1]{\MakeLowercase{#1}}
997 \DeclareRobustCommand*\FB@up@fake}[1]{%
998   \settoheight{\FB@Mht}{M}%
999   \addtolength{\FB@Mht}{\FBsupR \FB@Mht}%
1000   \addtolength{\FB@Mht}{-\FBsupS ex}%
1001   \raisebox{\FB@Mht}{\scalefont{\FBsupS}{\FB@lc{#1}}}%
1002   }

```

The only packages I currently know to take advantage of real superscripts are a) `realscripts` used in conjunction with XeLaTeX or LuaLaTeX and OpenType fonts having the font feature ‘VerticalPosition=Superior’ and b) `fourier` (from version 1.6) when Expert Utopia fonts are available.

`\FB@up` checks whether the current font is a Type1 ‘Expert’ (or ‘Pro’) font with real superscripts or not (the code works currently only with `fourier-1.6` but could work with any Expert Type1 font with built-in superscripts, see below), and decides to use real or fake superscripts. It works as follows: the content of `\f@family` (family name of the current font) is split by `\FB@split` into two pieces, the first three characters (‘fut’ for Fourier, ‘ppl’ for Adobe’s Palatino, ...) stored in `\FB@firstthree` and the rest stored in `\FB@suffix` which is expected to be ‘x’ or ‘j’ for expert fonts.

```

1003 \def\FB@split#1#2#3#4@nil{\def\FB@firstthree{#1#2#3}%
1004                               \def\FB@suffix{#4}}
1005 \def\FB@x{x}
1006 \def\FB@j{j}
1007 \DeclareRobustCommand*\FB@up}[1]{%
1008   \bgroup \FB@poormantrue
1009   \expandafter\FB@split\f@family@nil

```

Then `\FB@up` looks for a `.fd` file named `t1fut-sup.fd` (Fourier) or `t1ppl-sup.fd` (Palatino), etc. supposed to define the subfamily (fut-sup or ppl-sup, etc.) giving access to the built-in superscripts. If the `.fd` file is not found by `\IfFileExists`, `\FB@up` falls back on fake superscripts, otherwise `\FB@suffix` is checked to decide whether to use fake or real superscripts.

```

1010   \edef\reserved@a{\lowercase{%
1011     \noexpand\IfFileExists{\f@encoding\FB@firstthree -sup.fd}}}%
1012   \reserved@a
1013   {\ifx\FB@suffix\FB@x \FB@poormanfalse\fi
1014     \ifx\FB@suffix\FB@j \FB@poormanfalse\fi
1015     \ifFB@poorman \FB@up@fake{#1}%
1016     \else \FB@up@real{#1}%
1017     \fi}%
1018   {\FB@up@fake{#1}}%
1019   \egroup}

```

`\FB@up@real` just picks up the superscripts from the subfamily (and forces lowercase).

```

1020 \newcommand*\FB@up@real}[1]{\bgroup

```

```

1021     \fontfamily{\FB@firstthree -sup}\selectfont \FB@lc{#1}\egroup}
\fbup is defined as \FB@up unless \realsuperscript is defined by realscripts.sty.
\fbup just prints its argument in bookmarks.
1022 \DeclareRobustCommand*\fbup[1]{%
1023   \texorpdfstring{\ifx\realsuperscript\@undefined
1024     \FB@up{#1}%
1025   \else
1026     \bgroup\let\fakesuperscript\FB@up@fake
1027     \realsuperscript{\FB@lc{#1}}\egroup
1028   \fi
1029   }{#1}%
1030 }

```

Let's provide a temporary definition for \up (redefined 'AtBeginDocument' as \fbup or \textsuperscript according to \frenchsetup options).

```

1031 \providecommand*\up{\fbup}
Poor man's definition of \up for Plain.
1032 \else
1033 \providecommand*\up[1]{\leavevmode\raise1ex\hbox{\sevenrm #1}}
1034 \fi

```

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

```

\ier 1035 \def\ieme{\up{e}\xspace}
\iere 1036 \def\iemes{\up{es}\xspace}
\iemes 1037 \def\ier{\up{er}\xspace}
\iers 1038 \def\iers{\up{ers}\xspace}
\ieres 1039 \def\iere{\up{re}\xspace}
1040 \def\ieres{\up{res}\xspace}

```

```

\FBmedkern
\FBthickkern 1041 \newcommand*\FBmedkern{\kern+.2em}
1042 \newcommand*\FBthickkern{\kern+.3em}

```

`\primo` Some support macros relying on \up for numbering,

```

\fprimo) 1043 \newcommand*\FrenchEnumerate[1]{%
\ nos 1044 #1\texorpdfstring{\up{o}\FBthickkern}{\textdegree\space}}
\Nos 1045 \newcommand*\FrenchPopularEnumerate[1]{%
\No 1046 #1\texorpdfstring{\up{o})\FBthickkern}{\textdegree\space}}

```

`\no` Typing `\primo` should result in '°' (except in bookmarks where `\textdegree` is used instead of o-superior),

```

1047 \def\primo{\FrenchEnumerate1}
1048 \def\secundo{\FrenchEnumerate2}
1049 \def\tertio{\FrenchEnumerate3}
1050 \def\quarto{\FrenchEnumerate4}

```

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

```
1051 \def\fprimo){\FrenchPopularEnumerate1}
1052 \def\fsecundo){\FrenchPopularEnumerate2}
1053 \def\ftertio){\FrenchPopularEnumerate3}
1054 \def\fquarto){\FrenchPopularEnumerate4}
```

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

```
1055 \DeclareRobustCommand*\No}{%
1056   \texorpdfstring{N\up{o}\FBmedkern}{N\textdegree\space}}
1057 \DeclareRobustCommand*\no}{%
1058   \texorpdfstring{n\up{o}\FBmedkern}{n\textdegree\space}}
1059 \DeclareRobustCommand*\Nos}{%
1060   \texorpdfstring{N\up{os}\FBmedkern}{N\textdegree\space}}
1061 \DeclareRobustCommand*\nos}{%
1062   \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 avoidind hyphenation. A `\kern0pt` is used instead of `\mbox` because `\mbox` would break microtype’s font expansion; as a positive side effect, composed names (such as Dupont-Durand) can now be hyphenated on explicit hyphens.

```
1063 \ifLaTeXe
1064   \DeclareRobustCommand*\bname}[1]{%
1065     \texorpdfstring{\leavevmode\beginngroup\kern0pt #1\endngroup}{#1}}%
1066   }
1067   \DeclareRobustCommand*\bsc}[1]{%
1068     \texorpdfstring{\leavevmode\beginngroup\kern0pt \scshape #1\endngroup}%
1069       {\textsc{#1}}}%
1070   }
1071 \else
1072   \newcommand*\bname}[1]{\leavevmode\beginngroup\kern0pt #1\endngroup}
1073   \let\bsc\bname
1074 \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`. `\degree` can be accessed by the command `\r{}` for ring accent.

```
1075 \iffBunicode
1076   \providecommand*\textbackslash}{\char"005C}
1077   \providecommand*\textasciicircum}{\char"005E}
1078   \providecommand*\textasciitilde}{\char"007E}
1079   \DeclareRobustCommand*\degree}{°}
1080 \else
1081   \DeclareRobustCommand*\degree}{\textdegree}
```

```

1082 \fi
1083 \DeclareRobustCommand*\boi{\textbackslash}
1084 \DeclareRobustCommand*\circonflexe{\textasciicircum}
1085 \DeclareRobustCommand*\tild{\textasciitilde}
1086 \newcommand*\at{}{}

```

`\degrees` We now define a macro `\degrees` for typesetting the abbreviation for ‘degrees’ (as in ‘°C’ or ‘°K’) in text fonts which also works in math mode for angles.

```

1087 \DeclareRobustCommand*\degrees{\degree}
1088 \ifLaTeXe
1089   \AtBeginDocument{%
1090     \@ifpackageloaded{fontspec}{%
1091       {\DeclareRobustCommand*\degrees}{%
1092         \texorpdfstring{\hbox{\UseTextSymbol{TS1}{\textdegree}}}{%
1093           {\textdegree}}}%
1094       }%
1095     }
1096 \fi

```

2.6 Formatting numbers

`\StandardMathComma` As mentioned in the `TEXbook` 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 (or Acadian) *only* (no space added); `\StandardMathComma` switches back to the standard behaviour of the comma. Unfortunately, `\newcount` inside `\if` breaks Plain formats.

```

1097 \newif\ifFB@icomma
1098 \newcount\mc@charclass
1099 \newcount\mc@charfam
1100 \newcount\mc@charslot
1101 \newcount\std@mcc
1102 \newcount\dec@mcc
1103 \ifFBLuaTeX
1104   \mc@charclass=\Umathcharclass`,
1105   \newcommand*\dec@math@comma}{%
1106     \mc@charfam=\Umathcharfam`,
1107     \mc@charslot=\Umathcharslot`,
1108     \Umathcode`,= 0 \mc@charfam \mc@charslot
1109   }
1110   \newcommand*\std@math@comma}{%
1111     \mc@charfam=\Umathcharfam`,
1112     \mc@charslot=\Umathcharslot`,
1113     \Umathcode`,= \mc@charclass \mc@charfam \mc@charslot

```

```

1114 }
1115 \else
1116 \std@mcc=\mathcode`,
1117 \dec@mcc=\std@mcc
1118 \@tempcnta=\std@mcc
1119 \divide\@tempcnta by "1000
1120 \multiply\@tempcnta by "1000
1121 \advance\dec@mcc by -\@tempcnta
1122 \newcommand*\{dec@math@comma}\{mathcode`,=\dec@mcc}
1123 \newcommand*\{std@math@comma}\{mathcode`,=\std@mcc}
1124 \fi
1125 \let\dec@m@c\relax

```

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

```

1126 \newif\if@FBpreamble
1127 \ifLaTeXe \@FBpreambletrue \fi
1128 \newif\if@preamble@DecimalMathComma
1129 \newcommand*\{DecimalMathComma}\{%
1130 \if@FBpreamble \@preamble@DecimalMathCommatrue
1131 \else
1132 \ifFB@icomma
1133 \PackageWarning{french.ldf}\{%
1134 icomma package loaded, \protect\DecimalMathComma\MessageBreak
1135 does nothing. Reported}\%
1136 \else
1137 \ifFBfrench
1138 \dec@math@comma
1139 \let\dec@m@c\dec@math@comma
1140 \expandafter\addto\csname extras\language\name\endcsname
1141 {\dec@m@c}\%
1142 \fi
1143 \fi
1144 \fi
1145 }
1146 \newcommand*\{StandardMathComma}\{%
1147 \ifFB@icomma
1148 \PackageWarning{french.ldf}\{%
1149 icomma package loaded, \protect\StandardMathComma\MessageBreak
1150 does nothing. Reported}\%
1151 \else
1152 \ifFBfrench
1153 \std@math@comma
1154 \let\dec@m@c\relax
1155 \fi
1156 \fi

```

```
1157 }
```

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

```
1158 \ifLaTeXe\else
1159   \addto\noextrasfrench{\std@math@comma}
1160 \fi
```

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

```
1161 \newcommand*{\nombre}[1][{\#1}\fb@warning{*** \noexpand\nombre
1162                               no longer formats numbers\string! ***}]
```

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

```
1163 \ifFB@luatex@punct
1164   \activate@luatexpunct
1165 \fi
1166 \let\FBstop@here\relax
1167 \def\FBclean@on@exit{%
1168   \let\ifLaTeXe\iffalse
1169   \let\LaTeXtrue\undefined
1170   \let\LaTeXefalse\undefined
1171   \let\FB@llc\loadlocalcfg
1172   \let\loadlocalcfg@gobble}
1173 \ifx\magnification\@undefined
1174 \else
1175   \def\FBstop@here{%
1176     \FBclean@on@exit
1177     \ldf@finish\CurrentOption
1178     \let\loadlocalcfg\FB@llc
1179     \endinput}
1180 \fi
1181 \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 dialect. Can be canceled anytime by `\StandardMathComma`.

```
1182 \AtBeginDocument{%
1183   \@FBpreamblefalse
1184   \@ifpackageloaded{icomma}%
1185     {\FB@icommatrue
1186       \if@preamble@DecimalMathComma
1187         \PackageWarning{french.ldf}{%
1188           icomma package loaded, \protect\DecimalMathComma%
1189           \MessageBreak does nothing. Reported}%
1190     } \fi
```

```

1191 }%
1192 {\if@preamble@DecimalMathComma
1193 \ifFB@mainlanguage@FR \dec@math@comma \fi
1194 \let\dec@m@c\dec@math@comma
1195 \addto\extrasfrench{\dec@m@c}%
1196 \ifdefined\extrasacadian
1197 \addto\extrasacadian{\dec@m@c}%
1198 \fi
1199 \fi

```

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

```

1200 \addto\noextrasfrench{\std@math@comma}%
1201 \ifdefined\noextrasacadian
1202 \addto\noextrasacadian{\std@math@comma}%
1203 \fi
1204 }%
1205 }

```

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.

```

1206 \renewcommand*{\nombre}[1]{\Warning@nombre{#1}}
1207 \newcommand*{\Warning@nombre}[1]{%
1208 \ifdefined\numprint
1209 \numprint{#1}%
1210 \else
1211 \PackageWarning{french.ldf}{%
1212 \protect\nombre\space now relies on package numprint.sty,%
1213 \MessageBreak add \protect
1214 \usepackage[autolanguage]{numprint},\MessageBreak
1215 see file numprint.pdf for more options.\MessageBreak
1216 \protect\nombre\space called}%
1217 \global\let\Warning@nombre\relax
1218 {#1}%
1219 \fi
1220 }

1221 \newcommand*{\FBthousandsep}{\kern \fontdimen2\font \relax}

```

2.7 Caption names

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

`\captionfrench` Let's first define `\captionfrench` which sets all strings used in the four standard document classes provided with LaTeX.

`\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`.

```
1222 \providecommand*\FBfigtabshape{\scshape}
```

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

```
1223 \StartBabelCommands*\BabelLanguages}{captions}
1224     [unicode, fontenc=TU EU1 EU2, charset=utf8]
1225     \SetString{\refname}{Références}
1226     \SetString{\abstractname}{Résumé}
1227     \SetString{\prefacename}{Préface}
1228     \SetString{\contentsname}{Table des matières}
1229     \SetString{\ccname}{Copie à }
1230     \SetString{\proofname}{Démonstration}
1231     \SetString{\partfirst}{Première}
1232     \SetString{\partsecond}{Deuxième}
1233     \SetStringLoop{ordinal#1}{%
1234         \frenchpartfirst,\frenchpartsecond,Troisième,Quatrième,%
1235         Cinquième,Sixième,Septième,Huitième,Neuvième,Dixième,Onzième,%
1236         Douzième,Treizième,Quatorzième,Quinzième,Seizième,%
1237         Dix-septième,Dix-huitième,Dix-neuvième,Vingtième}
1238 \StartBabelCommands*\BabelLanguages}{captions}
1239     \SetString{\refname}{R\`eferences}
1240     \SetString{\abstractname}{R\`esum\`e}
1241     \SetString{\bibname}{Bibliographie}
1242     \SetString{\prefacename}{Pr\`eface}
1243     \SetString{\chaptername}{Chapitre}
1244     \SetString{\appendixname}{Annexe}
1245     \SetString{\contentsname}{Table des mati\`eres}
1246     \SetString{\listfigurename}{Table des figures}
1247     \SetString{\listtablename}{Liste des tableaux}
1248     \SetString{\indexname}{Index}
1249     \SetString{\figurename}{Figure}
1250     \SetString{\tablename}{Table}
1251     \SetString{\pagename}{page}
1252     \SetString{\seename}{voir}
1253     \SetString{\alsoname}{voir aussi}
1254     \SetString{\enclname}{P.~J. }
1255     \SetString{\ccname}{Copie \`a }
1256     \SetString{\headtoname}{}
1257     \SetString{\proofname}{D\`emonstration}
1258     \SetString{\glossaryname}{Glossaire}
```

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`.

```
1259 \SetString{\partfirst}{Premi\`ere}
1260 \SetString{\partsecond}{Deuxi\`eme}
1261 \SetString{\partnameord}{partie}
1262 \SetStringLoop{ordinal#1}{%
1263   \partfirst,\partsecond,Troisi\`eme,Quatri\`eme, Cinqui\`eme,%
1264   Sixi\`eme,Septi\`eme,Huiti\`eme,Neuvi\`eme,Dixi\`eme,%
1265   Onzi\`eme,Douzi\`eme,Treizi\`eme,Quatorzi\`eme,Quinzi\`eme,%
1266   Seizi\`eme,Dix-septi\`eme,Dix-huiti\`eme,Dix-neuvi\`eme,%
1267   Vingti\`eme}
1268 \AfterBabelCommands{%
1269   \DeclareRobustCommand*{\FB@emptypart}{\def\thepart{\unskip}}%
1270   \DeclareRobustCommand*{\FB@partname}{%
1271     \ifFBPartNameFull
1272       \csname ordinal\romannumeral\value{part}\endcsname\space
1273       \partnameord\FB@emptypart
1274     \else
1275       Partie%
1276     \fi}%
1277   }
1278   \SetString{\partname}{\FB@partname}
1279 \EndBabelCommands
```

`\figurename` and `\tablename` no longer include font commmands; to print them in small caps in French (the default), we now customise `\fnum@figure` and `\fnum@table` when available (not in `beamer.cls` f.i.).

```
1280 \AtBeginDocument{%
1281   \ifx\FBfigtabshape\relax
1282   \else
1283     \ifdefined\fnum@figure
1284       \let\fnum@figureORI\fnum@figure
1285       \renewcommand{\fnum@figure}{\ifFBfrench\FBfigtabshape\fi
1286                                     \fnum@figureORI}}%
1287     \fi
1288     \ifdefined\fnum@table
1289       \let\fnum@tableORI\fnum@table
1290       \renewcommand{\fnum@table}{\ifFBfrench\FBfigtabshape\fi
1291                                     \fnum@tableORI}}%
1292     \fi
1293   \fi
1294 }
```

2.8 Figure and table captions

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

```
1295 \newcommand{\FBWarning}[1]{\PackageWarning{french.ldf}{#1}}
```

`\CaptionSeparator` Let's consider now captions in figures and tables. In French, captions in figures and tables should never be printed as 'Figure 1:' which is the default in standard LaTeX2e classes (a space should precede the colon in French). This flaw may occur with pdfLaTeX as ':' is made active too late. With LuaLaTeX and XeLaTeX, this glitch doesn't occur, you get 'Figure 1 : ' which is correct in French. With pdfLaTeX `babel-french` provides the following workaround.

The standard definition of `\@makecaption` (e.g., the one provided in `article.cls`, `report.cls`, `book.cls` which is frozen for LaTeX2e according to Frank Mittelbach), is saved in `\STD@makecaption`. 'AtBeginDocument' we compare it to its current definition (some classes like `memoir`, `koma-script` classes, `AMS` classes, `ua-thesis.cls`... change it). If they are identical, `babel-french` just adds a hook called `\FBCaption@Separator` to `\@makecaption`; `\FBCaption@Separator` defaults to ':' as in the standard definition of `\@makecaption` and will be changed to ' :' in French 'AtBeginDocument'; it can be also set to `\CaptionSeparator` (' - ') using `CustomiseFigTabCaptions`.

While saving the standard definition of `\@makecaption` we have to make sure that characters ':' and '>' have `\catcode 12` (`babel-french` makes ':' active and `spanish.lfd` makes '>' active).

```
1296 \bgroup
1297 \catcode`:=12 \catcode`>=12 \relax
1298 \long\gdef\STD@makecaption#1#2{%
1299   \vskip\abovecaptionskip
1300   \sbox\@tempboxa{#1: #2}%
1301   \ifdim \wd\@tempboxa >\hsize
1302     #1: #2\par
1303   \else
1304     \global \@minipagefalse
1305     \hb@xt@\hsize{\hfil\box\@tempboxa\hfil}%
1306     \fi
1307   \vskip\belowcaptionskip}
1308 \egroup
```

No warning is issued for SMF and AMS classes as their layout of captions is compatible with French typographic standards.

With `memoir` and `koma-script` classes, `babel-french` customises `\captiondelim` or `\captionformat` in French (unless option `CustomiseFigTabCaptions` is set to `false`) and issues no warning.

When `\@makecaption` has been changed by another class or package, a warning is printed in the `.log` file.

Enable the standard warning only if high punctuation is active.

```

1309 \newif\if@FBwarning@capsep
1310 \ifFB@active@punct\@FBwarning@capseptrue\fi
1311 \newcommand*{\CaptionSeparator}{\space\textendash\space}
1312 \def\FBCaption@Separator{: }
1313 \long\def\FB@makecaption#1#2{%
1314   \vskip\abovecaptionskip
1315   \sbox\@tempboxa{#1\FBCaption@Separator #2}%
1316   \ifdim \wd\@tempboxa >\hsize
1317     #1\FBCaption@Separator #2\par
1318   \else
1319     \global \@minipagefalse
1320     \hb@xt@\hsize{\hfil\box\@tempboxa\hfil}%
1321   \fi
1322   \vskip\belowcaptionskip}

```

Disable the standard warning with AMS and SMF classes.

```

1323 \@ifclassloaded{amsart}{\@FBwarning@capsepfalse}{}
1324 \@ifclassloaded{amsbook}{\@FBwarning@capsepfalse}{}
1325 \@ifclassloaded{amsdtx}{\@FBwarning@capsepfalse}{}
1326 \@ifclassloaded{amslatex}{\@FBwarning@capsepfalse}{}
1327 \@ifclassloaded{amproc}{\@FBwarning@capsepfalse}{}
1328 \@ifclassloaded{smfart}{\@FBwarning@capsepfalse}{}
1329 \@ifclassloaded{smfbook}{\@FBwarning@capsepfalse}{}

```

Disable the standard warning for some classes that do not use ‘:’ as caption separator.

```

1330 \@ifclassloaded{IEEEconf}{\@FBwarning@capsepfalse}{}
1331 \@ifclassloaded{IEEEtran}{\@FBwarning@capsepfalse}{}
1332 \@ifclassloaded{revtex4-2}{\@FBwarning@capsepfalse}{}
1333 \@ifclassloaded{svjour3}{\@FBwarning@capsepfalse}{}

```

No warning with memoir or koma-script classes: they change `\@makecaption` but we will manage to customise them in French later on (see below after executing `\FBprocess@options`)

```

1334 \@ifclassloaded{memoir}{\@FBwarning@capsepfalse}{}
1335 \ifFB@koma \@FBwarning@capsepfalse \fi

```

No warning with the beamer class which defines `\beamer@makecaption` (customised below) instead of `\@makecaption`. No warning either if `\@makecaption` is undefined (i.e. letter).

```

1336 \@ifclassloaded{beamer}{\@FBwarning@capsepfalse}{}
1337 \ifdefined\@makecaption\else\@FBwarning@capsepfalse\fi

```

First check the definition of `\@makecaption`, change it or issue a warning in case it has been changed by a class or package not (yet) compatible with `babel-french`; then change the definition of `\CaptionSeparator`, taking care that the colon is typeset correctly in French (*not* ‘Figure 1: légende’).

```

1338 \AtBeginDocument{%
1339   \ifx\@makecaption\STD@makecaption

```

```

1340 \global\let\@makecaption\FB@makecaption
If OldFigTabCaptions=true, do not overwrite \FBCaption@Separator (already saved
as ‘:’ for other languages and set to \CaptionSeparator by \extrasfrench when
French is the main language); otherwise locally force \autospace@beforeFDP in case
AutoSpacePunctuation=false.
1341 \ifFBOldFigTabCaptions
1342 \else
1343 \def\FBCaption@Separator{\autospace@beforeFDP : }%
1344 \ifBFCustomiseFigTabCaptions
1345 \ifFB@mainlanguage@FR
1346 \def\FBCaption@Separator{\CaptionSeparator}%
1347 \fi
1348 \fi
1349 \fi
1350 \@FBwarning@capsepfalse
1351 \fi
No Warning if caption.sty or caption-light.sty has been loaded.
1352 \@ifpackageloaded{caption}{\@FBwarning@capsepfalse}%
1353 \@ifpackageloaded{caption-light}{\@FBwarning@capsepfalse}%
Final warning if relevant:
1354 \if@FBwarning@capsep
1355 \FBWarning
1356 {Figures' and tables' captions might look like\MessageBreak
1357 `Figure 1:' in French instead of `Figure 1 :'.\MessageBreak
1358 If this happens, to fix this issue\MessageBreak
1359 switch to LuaLaTeX or XeLaTeX or\MessageBreak
1360 try to add \protect\usepackage{caption} or\MessageBreak
1361 ... leave it as it is; reported}%
1362 \fi
1363 \let\FB@makecaption\relax
1364 \let\STD@makecaption\relax
1365 }

```

2.9 Dots...

`\FBtextellipsis` Unless a ready-made character is available in the current font, LaTeX’s default definition of `\textellipsis` includes a `\kern` at the end; this space is not wanted in some cases (before a closing brace for instance) and `\kern` breaks hyphenation of the next word. We define `\FBtextellipsis` for French (in LaTeX only) the same way but without the last `\kern`.

LY1 has a ready made character for `\textellipsis`, it should be used in French. The same is true for Unicode fonts in use with XeTeX and LuaTeX.

```
1366 \ifFBunicode
```

```

1367 \else
1368   \DeclareTextSymbol{\FBtextellipsis}{LY1}{133}
1369   \DeclareTextCommand{\FBtextellipsis}{PU}{\9040\046}
1370   \DeclareTextCommand{\FBtextellipsis}{PD1}{\203}
1371   \DeclareTextCommandDefault{\FBtextellipsis}{%
1372     .\kern\fontdimen3\font.\kern\fontdimen3\font.\xspace}%
1373   \def\bbl@frenchdots{\babel@save\textellipsis
1374     \let\textellipsis\FBtextellipsis}
1375   \addto\extrasfrench{\bbl@frenchdots}
1376 \fi

```

2.10 More checks about packages' loading order

Like packages `captions` and `floatrow` (see section 2.8), package listings should be loaded after `babel-french` due to active characters issues (pdfLaTeX only).

```

1377 \ifFB@active@punct
1378   \@ifpackageloaded{listings}
1379     {\AtBeginDocument{%
1380       \FBWarning{Please load the "listings" package\MessageBreak
1381         AFTER babel/french; reported}}%
1382     }{}
1383 \fi

```

Package `natbib` should be loaded before `babel-french` due to active characters issues (pdfLaTeX only).

```

1384 \newif\if@FBwarning@natbib
1385 \ifFB@active@punct
1386   \@ifpackageloaded{natbib}{}{\@FBwarning@natbibtrue}
1387 \fi
1388 \AtBeginDocument{%
1389   \if@FBwarning@natbib
1390     \@ifpackageloaded{natbib}{}{\@FBwarning@natbibfalse}%
1391   \fi
1392   \if@FBwarning@natbib
1393     \FBWarning{Please load the "natbib" package\MessageBreak
1394       BEFORE babel/french; reported}%
1395   \fi
1396 }

```

Package `beamerarticle` should be loaded before `babel-french` to avoid list's conflicts, see p. 60.

```

1397 \newif\if@FBwarning@beamerarticle
1398 \@ifpackageloaded{beamerarticle}{}{\@FBwarning@beamerarticlettrue}
1399 \AtBeginDocument{%
1400   \if@FBwarning@beamerarticle
1401     \@ifpackageloaded{beamerarticle}{}%

```

```

1402                                     {\@FBwarning@beamerarticlefalse}%
1403 \fi
1404 \if@FBwarning@beamerarticle
1405   \FBWarning{Please load the "beamerarticle" package\MessageBreak
1406             BEFORE babel/french; reported}%
1407 \fi
1408 }

```

2.11 Setup options: key/value stuff (l3keys)

Check LaTeX2e version (support for l3keys required).

```
1409 \NeedsTeXFormat{LaTeX2e}[2022-06-01]
```

All setup options are handled by command `\frenchsetup{}` based on the l3keys' `\SetKeys{}` command. A list of flags is defined beforehand and set to default values which will possibly be changed 'AtEndOfPackage' in case French is the main language. After this, `\frenchsetup{}` eventually modifies the preset values of these flags.

Some options processing occurs in `\frenchsetup{}`, *only for options explicitly set by `\frenchsetup{}`*, the rest 'AtBeginDocument'; any option affecting `\extrasfrench{}` *must* be immediately processed by `\frenchsetup{}`: when French is the main language, `\extrasfrench{}` is executed by Babel when it switches the main language and this occurs *before* reading the stuff postponed by babel-french 'AtBeginDocument'. Reexecuting `\extrasfrench{}` is not an option because of its side-effects (f.i. `\babel@save` and `\babel@savevariable` did not work for French).

We first define a collection of conditionals and set their defaults (true or false).

```

1410 \newif\ifFBShowOptions
1411 \newif\ifFBStandardLayout           \FBStandardLayouttrue
1412 \newif\ifFBGlobalLayoutFrench      \FBGlobalLayoutFrenchtrue
1413 \newif\ifFBStandardListSpacing     \FBStandardListSpacingtrue
1414 \newif\ifFBListOldLayout
1415 \newif\ifFBListItemsAsPar
1416 \newif\ifFBCompactItemize
1417 \newif\ifFBStandardItemizeEnv      \FBStandardItemizeEnvtrue
1418 \newif\ifFBStandardEnumerateEnv    \FBStandardEnumerateEnvtrue
1419 \newif\ifFBStandardItemLabels      \FBStandardItemLabelstrue
1420 \newif\ifFBStandardLists           \FBStandardListstrue
1421 \newif\ifFBIndentFirst
1422 \newif\ifFBFrenchFootnotes
1423 \newif\ifBFAutoSpaceFootnotes
1424 \newif\ifFBOriginalTypewriter
1425 \newif\ifFBThinColonSpace
1426 \newif\ifFBThinSpaceInFrenchNumbers
1427 \newif\ifFBFrenchSuperscripts      \FBFrenchSuperscriptstrue
1428 \newif\ifFBLowercaseSuperscripts   \FBLowercaseSuperscriptstrue
1429 \newif\ifFBPartNameFull             \FBPartNameFulltrue

```

```

1430 \newif\ifFBCustomiseFigTabCaptions
1431 \newif\ifFBOldFigTabCaptions
1432 \newif\ifFBSmallCapsFigTabCaptions \FBSmallCapsFigTabCaptionstrue
1433 \newif\ifFBSuppressWarning
1434 \newif\ifFBINGuillSpace

```

The following patch is for koma-script classes: the `\partformat` command, defined as `\partname~\thepart\autodot`, is incompatible with our redefinition of `\partname`.

```

1435 \ifFB@koma
1436 \ifdefined\partformat
1437 \def\FB@partformat@fix{%
1438     \iffBPartNameFull
1439     \babel@save\partformat
1440     \renewcommand*{\partformat}{\partname}%
1441     \fi}
1442 \addto\extrasfrench{\FB@partformat@fix}%
1443 \fi
1444 \fi

```

The default values of these flags are chosen so that `babel-french` does not change anything regarding the global layout. Some of them must be toggled when French (or a French dialect) 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` checking 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 2.10; a light customisation is compatible with the `beamerarticle` package.

```

1445 \def\FB@french{french}
1446 \def\FB@acadian{acadian}
1447 \newif\ifFB@mainlanguage@FR
1448 \AtEndOfPackage{%
1449     \ifx\bbl@main@language\FB@french \FB@mainlanguage@FRtrue
1450     \else \ifx\bbl@main@language\FB@acadian \FB@mainlanguage@FRtrue \fi
1451     \fi
1452     \ifFB@mainlanguage@FR
1453     \FBGlobalLayoutFrenchtrue
1454     \@ifclassloaded{beamer}%
1455     {\PackageInfo{french.1df}{%
1456         No list customisation for the beamer class,%
1457         \MessageBreak reported}}%
1458     {\@ifpackageloaded{beamerarticle}%
1459     {\FBStandardItemLabelsfalse
1460     \FBStandardListSpacingfalse

```

```

1461     \PackageInfo{french.ldf}{%
1462         Minimal list customisation for the beamerarticle%
1463         \MessageBreak package; reported}}%

```

Otherwise customise lists “à la française”:

```

1464     {\FBStandardListSpacingfalse
1465      \FBStandardItemizeEnvfalse
1466      \FBStandardEnumerateEnvfalse
1467      \FBStandardItemLabelsfalse}%
1468     }
1469     \FBIndentFirsttrue
1470     \FBFrenchFootnotestru
1471     \FBAutoSpaceFootnotestru
1472     \FBCustomiseFigTabCaptionstru
1473 \fi
1474 }

```

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

```

1475 \DeclareKeys[FBsetup]
1476 {
1477     ShowOptions.if           = FBShowOptions           ,
1478     StandardLayout.default:n = {true}           ,
1479     StandardLayout.code      = \FBStandardLayout@setup{#1} ,
1480     GlobalLayoutFrench.default:n = {true}           ,
1481     GlobalLayoutFrench.code  = \FBGlobalLayout@setup{#1}  ,
1482     StandardListSpacing.if   = FBStandardListSpacing    ,
1483     ReduceListSpacing.ifnot  = FBStandardListSpacing    ,
1484     ListOldLayout.default:n  = {true}           ,
1485     ListOldLayout.code       = \FBListOldLayout@setup{#1} ,
1486     CompactItemize.default:n = {true}           ,
1487     CompactItemize.code      = \FBCompactItemize@setup{#1} ,
1488     StandardItemizeEnv.if    = FBStandardItemizeEnv      ,
1489     StandardEnumerateEnv.if  = FBStandardEnumerateEnv  ,
1490     StandardItemLabels.if    = FBStandardItemLabels      ,
1491     ItemLabels.store         = \FrenchLabelItem          ,
1492     ItemLabeli.store         = \Frlabelitemi             ,
1493     ItemLabelii.store        = \Frlabelitemii            ,
1494     ItemLabeliii.store       = \Frlabelitemiii           ,
1495     ItemLabeliv.store        = \Frlabelitemiv            ,
1496     StandardLists.default:n  = {true}           ,
1497     StandardLists.code       = \FBStandardLists@setup{#1} ,
1498     ListItemsAsPar.if        = FBListItemsAsPar          ,
1499     IndentFirst.if           = FBIndentFirst             ,
1500     FrenchFootnotes.if      = FBFrenchFootnotes         ,
1501     AutoSpaceFootnotes.if   = FBAutoSpaceFootnotes      ,
1502     AutoSpacePunctuation.if = FBAutoSpacePunctuation    ,
1503     OriginalTypewriter.if    = FBOriginalTypewriter      ,

```



```

1504 ThinColonSpace.default:n = {true} ,
1505 ThinColonSpace.code = \FBThinColonSpace@setup{#1} ,
1506 ThinSpaceInFrenchNumbers.if = FBThinSpaceInFrenchNumbers ,
1507 FrenchSuperscripts.if = FBFrenchSuperscripts ,
1508 LowercaseSuperscripts.if = FBLowercaseSuperscripts ,
1509 PartNameFull.if = FBPartNameFull ,
1510 CustomiseFigTabCaptions.if = FBCustomiseFigTabCaptions ,
1511 OldFigTabCaptions.default:n = {true} ,
1512 OldFigTabCaptions.code = \FBOldFigTabCaptions@setup{#1} ,
1513 SmallCapsFigTabCaptions.default:n = {true} ,
1514 SmallCapsFigTabCaptions.code = \FBSmallCapsFigTabCaptions@setup{#1} ,
1515 SuppressWarning.default:n = {true} ,
1516 SuppressWarning.code = \FBSuppressWarning@setup{#1} ,
1517 INGuillSpace.default:n = {true} ,
1518 INGuillSpace.code = \FBINGuillSpace@setup{#1} ,
1519 InnerGuillSingle.if = FBInnerGuillSingle ,
1520 EveryParGuill.default:n = {open} ,
1521 EveryParGuill.code = \FBEveryParGuill@setup{#1} ,
1522 EveryLineGuill.default:n = {open} ,
1523 EveryLineGuill.code = \FBEveryLineGuill@setup{#1} ,
1524 UnicodeNoBreakSpaces.default:n = {true} ,
1525 UnicodeNoBreakSpaces.code = \FBUnicodeNoBreakSpaces@setup{#1} ,
1526 og.code = \FBog@setup{#1} ,
1527 fg.code = \FBfg@setup{#1} ,
1528 }

```

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

```

1529 \newcommand*{\frenchsetup}[1]{%
1530   \SetKeys[FBsetup]{#1}%
1531 }%
1532 \@onlypreamble\frenchsetup

```

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

```

1533 \let\frenchbsetup\frenchsetup
1534 \@onlypreamble\frenchbsetup

```

The following commands, defined with property `.code` in `DeclareKeys{}`, execute some post-treatment required to immediately take the flags value into account.

```

1535 \newcommand*{\FBStandardLayout@setup}[1]%
1536   {\ifFB@mainlanguage@FR
1537     \csname FBStandardLayout#1\endcsname
1538   \else
1539     \PackageWarning{french.1df}%
1540     {Option `StandardLayout' skipped:\MessageBreak
1541     French is *not* babel's last option.\MessageBreak
1542     Reported}%

```

```

1543 \fi
1544 \ifFBStandardLayout
1545 \FBStandardListSpacingtrue
1546 \FBStandardItemizeEnvtrue
1547 \FBStandardItemLabelstrue
1548 \FBStandardEnumerateEnvtrue
1549 \FBIndentFirstfalse
1550 \FBFrenchFootnotesfalse
1551 \FBAutoSpaceFootnotesfalse
1552 \else
1553 \FBStandardListSpacingfalse
1554 \FBStandardItemizeEnvfalse
1555 \FBStandardItemLabelsfalse
1556 \FBStandardEnumerateEnvfalse
1557 \FBIndentFirsttrue
1558 \FBFrenchFootnotesttrue
1559 \FBAutoSpaceFootnotesttrue
1560 \fi
1561 }
1562 \newcommand*\FBGlobalLayout@setup}[1]%
1563 {\ifFB@mainlanguage@FR
1564 \csname FBGlobalLayoutFrench#1\endcsname
1565 \else
1566 \PackageWarning{french.1df}%
1567 {Option `GlobalLayoutFrench' skipped:\MessageBreak
1568 French is *not* babel's last option.\MessageBreak
1569 Reported}%
1570 \fi
1571 }
1572 \newcommand*\FB@ListOldLayout@setup}[1]%
1573 {\csname FBListOldLayout#1\endcsname
1574 \ifFBListOldLayout
1575 \FBStandardEnumerateEnvtrue
1576 \renewcommand*\FrenchLabelItem}{\textendash}%
1577 \fi
1578 }
1579 \newcommand*\FB@CompactItemize@setup}[1]%
1580 {\csname FBCompactItemize#1\endcsname
1581 \ifFBCompactItemize
1582 \FBStandardItemizeEnvfalse
1583 \FBStandardEnumerateEnvfalse
1584 \else
1585 \FBStandardItemizeEnvtrue
1586 \FBStandardEnumerateEnvtrue
1587 \fi
1588 }

```

```

1589 \newcommand*{\FBStandardLists@setup}[1]%
1590   {\csname FBStandardLists#1\endcsname
1591     \ifFBStandardLists
1592       \FBStandardListSpacingtrue
1593       \FBStandardItemizeEnvtrue
1594       \FBStandardEnumerateEnvtrue
1595       \FBStandardItemLabelstrue
1596     \else
1597       \FBStandardListSpacingfalse
1598       \FBStandardItemizeEnvfalse
1599       \FBStandardEnumerateEnvfalse
1600       \FBStandardItemLabelsfalse
1601     \fi
1602   }
1603 \newcommand*{\FBThinColonSpace@setup}[1]%
1604   {\csname FBThinColonSpace#1\endcsname
1605     \ifFBThinColonSpace
1606       \renewcommand*{\FBcolonspace}{\FBthinspace}%
1607     \fi
1608   }
1609 \newcommand*{\FBOldFigTabCaptions@setup}[1]%
1610   {\csname FBOldFigTabCaptions#1\endcsname
1611     \ifFBOldFigTabCaptions
1612       \def\FB@capsep@fix{\babel@save\FBCaption@Separator
1613         \def\FBCaption@Separator{\CaptionSeparator}}%
1614       \addto\extrasfrench{\FB@capsep@fix}%
1615       \ifdefined\extrasacadian
1616         \addto\extrasacadian{\FB@capsep@fix}%
1617       \fi
1618     \fi
1619   }
1620 \newcommand*{\FBSmallCapsFigTabCaptions@setup}[1]%
1621   {\csname FBSmallCapsFigTabCaptions#1\endcsname
1622     \ifFBSmallCapsFigTabCaptions
1623     \else
1624       \let\FBfigtabshape\relax
1625     \fi
1626   }
1627 \newcommand*{\FBSuppressWarning@setup}[1]%
1628   {\csname FBSuppressWarning#1\endcsname
1629     \ifFBSuppressWarning
1630       \renewcommand{\FBWarning}[1]{}%
1631     \fi
1632   }
1633 \newcommand*{\FBINGuillSpace@setup}[1]%
1634   {\csname FBINGuillSpace#1\endcsname

```

```

1635 \ifFBINGuillSpace
1636 \renewcommand*{\FBguillspace}{\space}%
1637 \fi
1638 }
1639 \newcommand*{\FBEveryParGuill@setup}[1]%
1640 {\expandafter\let\expandafter
1641 \FBeveryparguill\csname FBguill#1\endcsname
1642 \ifx\FBEveryparguill\FBguillopen
1643 \else\ifx\FBEveryparguill\FBguillclose
1644 \else\ifx\FBEveryparguill\FBguillnone
1645 \else
1646 \let\FBEveryparguill\FBguillopen
1647 \FBWarning{Wrong value for `EveryParGuill':
1648 try `open',\MessageBreak
1649 `close' or `none'. Reported}%
1650 \fi
1651 \fi
1652 \fi
1653 }
1654 \newcommand*{\FBEveryLineGuill@setup}[1]%
1655 {\ifFB@luatex@punct
1656 \expandafter\let\expandafter
1657 \FBeverylineguill\csname FBguill#1\endcsname
1658 \ifx\FBEverylineguill\FBguillopen
1659 \else\ifx\FBEverylineguill\FBguillclose
1660 \else\ifx\FBEverylineguill\FBguillnone
1661 \else
1662 \let\FBEverylineguill\FBguillnone
1663 \FBWarning{Wrong value for `EveryLineGuill':
1664 try `open',\MessageBreak
1665 `close' or `none'. Reported}%
1666 \fi
1667 \fi
1668 \fi
1669 \else
1670 \FBWarning{Option `EveryLineGuill' skipped:%
1671 \MessageBreak this option is for
1672 LuaTeX *only*.\MessageBreak Reported}%
1673 \fi
1674 }

```

Option **UnicodeNoBreakSpaces** (LuaLaTeX only) 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.

```

1675 \newcommand*{\FBUnicodeNoBreakSpaces@setup}[1]%
1676 {\ifFB@luatex@punct
1677 \csname FBucsNBSP#1\endcsname

```

```

1678     \ifFBucsNBSP \FB@ucsNBSP=\@ne \fi
1679   \else
1680     \FBWarning{Option `UnicodeNoBreakSpaces' skipped:%
1681               \MessageBreak this option is for
1682               LuaTeX *only*.\MessageBreak Reported}%
1683   \fi
1684 }%

```

Inputting French quotes as *single characters* when they are available on the keyboard (through a compose key for instance) is more comfortable than typing `\og` and `\fg`. Life is simple here with modern LuaTeX or XeTeX engines: we just have to activate the `\FB@addGUIspace` attribute for LuaTeX or set `\XeTeXcharclass` of quotes to the proper value for XeTeX.

With pdfTeX (or old LuaTeX and XeTeX engines), quote characters are made active and expand to `\og\ignorespaces` and `{\fg}` respectively if the current language is French, and to `\guillemotleft` and `\guillemotright` otherwise (think of German quotes), this is done by `\FB@@og` and `\FB@@fg`; thus correct non-breaking spaces will be added automatically to French quotes. The quote characters typed in depend on the input encoding, it can be single-byte (latin1, latin9, applemac,...) or multi-bytes (utf-8, utf8x); the next command is meant for checking whether a character is single-byte (`\FB@second` is empty) or not.

```

1685 \def\FB@parse#1#2\endparse{\def\FB@second{#2}}%
1686 \newcommand*{\FB@@og}%
1687   {\ifFBfrench
1688     \ifFB@spacing \FB@og\ignorespaces
1689     \else \guillemotleft
1690     \fi
1691   \else \guillemotleft
1692   \fi
1693 }
1694 \newcommand*{\FB@@fg}%
1695   {\ifFBfrench
1696     \ifFB@spacing \FB@fg
1697     \else \guillemotright
1698     \fi
1699   \else \guillemotright
1700   \fi
1701 }
1702 \newcommand*{\FBog@setup}[1]%
1703   {\ifFBunicode

```

LuaTeX or XeTeX in use, first try modern LuaTeX: we just need to set LuaTeX's attribute `\FB@addGUIspace` to 1,

```

1704   \ifFB@luatex@punct
1705     \FB@addGUIspace=1 \relax
1706   \fi

```

then with XeTeX it is a bit more tricky:

```
1707     \ifFB@xetex@punct
```

`\XeTeXinterchartokenstate` is defined, we just need to set `\XeTeXcharclass` to `\FB@guilo` for the French opening quote in T1 and Unicode encoding (see subsection 2.2).

```
1708     \XeTeXcharclass"13 = \FB@guilo
1709     \XeTeXcharclass"AB = \FB@guilo
1710     \XeTeXcharclass"A0 = \FB@guilnul
1711     \XeTeXcharclass"202F = \FB@guilnul
1712     \fi
```

Issue a warning with older Unicode engines requiring active characters.

```
1713     \ifFB@active@punct
1714         \FBWarning{Option og=« not supported with this version of
1715                 \MessageBreak LuaTeX/XeTeX; reported}% »
1716     \fi
1717 \else
```

This is for conventional TeX engines:

```
1718     \AtBeginDocument{%
1719         \ifdefined\uc@dc1c
Package inputenc with utf8x (ucs) encoding loaded, use \uc@dc1c:
1720         \uc@dc1c{171}{default}{\FB@@og}%
1721     \else
```

if encoding is not utf8x, check if the argument of og is a single-byte character:

```
1722         \FB@parse#1\endparse
1723         \ifx\FB@second\@empty
```

This means 8-bit character encoding. Package MULEenc (from CJK) defines `\mule@def` to map characters to control sequences.

```
1724         \ifdefined\mule@def
1725             \mule@def{11}{\FB@@og}%
1726         \else
1727             \ifdefined\DeclareInputText
1728                 \@tempcnta`#1\relax
1729                 \DeclareInputText{\the\@tempcnta}{\FB@@og}%
1730             \else
```

Package inputenc not loaded, no way...

```
1731             \FBWarning{Option `og' requires package
1732                     inputenc;\MessageBreak reported}%
1733         \fi
1734     \fi
1735 \else
```

This means multi-byte character encoding, we assume UTF-8

```
1736     \DeclareUnicodeCharacter{00AB}{\FB@@og}%
```

```

1737         \fi
1738     \fi}%
1739 \fi
1740 }

```

Same code for the closing quote.

```

1741 \newcommand*{\FBfg@setup}[1]%
1742 {\ifFBunicode
1743     \ifFB@luatex@punct
1744         \FB@addGUIspace=1 \relax
1745     \fi
1746     \ifFB@xetex@punct
1747         \XeTeXcharclass"14 = \FB@guilf
1748         \XeTeXcharclass"BB = \FB@guilf
1749         \XeTeXcharclass"A0 = \FB@guilnul
1750         \XeTeXcharclass"202F = \FB@guilnul
1751     \fi
1752     \ifFB@active@punct
1753         \FBWarning{Option fg=> not supported with this version of
1754             \MessageBreak LuaTeX/XeTeX; reported}%
1755     \fi
1756 \else
1757     \AtBeginDocument{%
1758         \ifdefined\uc@dclc
1759             \uc@dclc{187}{default}{\FB@@fg}%
1760         \else
1761             \FB@parse#1\endparse
1762             \ifx\FB@second\@empty
1763                 \ifdefined\mule@def
1764                     \mule@def{27}{\FB@@fg}%
1765                 \else
1766                     \ifdefined\DeclareInputText
1767                         \@tempcnta`#1\relax
1768                         \DeclareInputText{\the\@tempcnta}{\FB@@fg}%
1769                     \else
1770                         \FBWarning{Option `fg' requires package
1771                             inputenc;\MessageBreak reported}%
1772                     \fi
1773                 \fi
1774             \else
1775                 \DeclareUnicodeCharacter{00BB}{\FB@@fg}%
1776             \fi
1777         \fi}%
1778 \fi
1779 }
1780 % \end{macro}
1781 %

```

```

1782 % \begin{macro}{\FBprocess@options}
1783 %   |\FBprocess@options| will be executed at |\begin{document}|:
1784 %   it first checks about packages loaded in the preamble (possibly
1785 %   after \babel) which customise lists: currently \pkg{enumitem},
1786 %   \pkg{paralist} and \pkg{enumerate}; then it processes the options
1787 %   as set by \fbsetup{} or forced for compatibility with packages
1788 %   loaded in the preamble.
1789 %
1790 %   When French is the main language, |\extrasfrench| and
1791 %   |\captionsfrench| \emph{have already been processed} by \babel{}
1792 %   at |\begin{document}| \emph{before} |\FBprocess@options|.
1793 %   \begin{macrocode}
1794 \newcommand*{\FBprocess@options}{%

```

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

```

1795   \@ifpackageloaded{enumitem}{%
1796     \ifFBStandardItemizeEnv
1797     \else
1798       \FBStandardItemizeEnvtrue
1799       \PackageInfo{french.ldf}%
1800       {Setting StandardItemizeEnv=true for\MessageBreak
1801         compatibility with enumitem package,\MessageBreak
1802         reported}%
1803     \fi
1804     \ifFBStandardEnumerateEnv
1805     \else
1806       \FBStandardEnumerateEnvtrue
1807       \PackageInfo{french.ldf}%
1808       {Setting StandardEnumerateEnv=true for\MessageBreak
1809         compatibility with enumitem package,\MessageBreak
1810         reported}%
1811     \fi}}%
1812   \@ifpackageloaded{paralist}{%
1813     \ifFBStandardItemizeEnv
1814     \else
1815       \FBStandardItemizeEnvtrue
1816       \PackageInfo{french.ldf}%
1817       {Setting StandardItemizeEnv=true for\MessageBreak
1818         compatibility with paralist package,\MessageBreak
1819         reported}%
1820     \fi
1821     \ifFBStandardEnumerateEnv
1822     \else
1823       \FBStandardEnumerateEnvtrue
1824       \PackageInfo{french.ldf}%
1825       {Setting StandardEnumerateEnv=true for\MessageBreak

```



```

1826         compatibility with paralist package,\MessageBreak
1827         reported}%
1828     \fi}}}%
1829 \@ifpackageloaded{enumerate}{%
1830     \ifFBStandardEnumerateEnv
1831     \else
1832         \FBStandardEnumerateEnvtrue
1833         \PackageInfo{french.1df}%
1834         {Setting StandardEnumerateEnv=true for\MessageBreak
1835         compatibility with enumerate package,\MessageBreak
1836         reported}%
1837     \fi}}}%

```

Reset `\FB@ufl`'s normal meaning and update lists' settings now in case French is the main language:

```

1838 \def\FB@ufl{\update@frenchlists}
1839 \ifFB@mainlanguage@FR
1840     \update@frenchlists
1841 \else
1842     \ifFBStandardItemizeEnv
1843     \else
1844         \PackageWarning{french.1df}%
1845         {babel-french will not customise lists' layout\MessageBreak
1846         when French is not the main language,\MessageBreak
1847         reported}%
1848     \fi
1849 \fi

```

The layout of footnotes is handled at the `\begin{document}` depending on the values of flags `FrenchFootnotes` and `AutoSpaceFootnotes` (see section 2.14), nothing has to be done here for footnotes.

`AutoSpacePunctuation` adds a non-breaking space (in French only) before the four active characters (;:!?) even if none has been typed before them.

```

1850 \ifFBAutoSpacePunctuation
1851     \autospace@beforeFDP
1852 \else
1853     \noautospace@beforeFDP
1854 \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.

```

1855 \ifFBOriginalTypewriter
1856 \else
1857     \let\ttfamilyORI\ttfamily
1858     \let\rmfamilyORI\rmfamily

```

```

1859 \let\sffamilyORI\sffamily
1860 \let\ttfamily\ttfamilyFB
1861 \let\rmfamily\rmfamilyFB
1862 \let\sffamily\sffamilyFB
1863 \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.

```

1864 \@ifpackageloaded{numprint}%
1865   {\ifnprt@autolanguage
1866     \providecommand*\npstylefrench{}}%
1867     \iffBThinSpaceInFrenchNumbers
1868     \renewcommand*\FBthousandsep{\FBthinspace}%
1869   \fi
1870   \g@addto@macro\npstylefrench{\npthousandsep{\FBthousandsep}}%
1871 \fi
1872 }{}%

```

FrenchSuperscripts: if `true` `\up=\fup`, else `\up=\textsuperscript`. The star-form `\up*=\FB@up@fake` is provided for fonts that lack some superior letters: Adobe Jenson Pro and Utopia Expert have no “g superior” for instance.

```

1873 \iffBFrenchSuperscripts
1874   \DeclareRobustCommand*\up{}%
1875   \texorpdfstring{\@ifstar{\FB@up@fake}{\fup}}{}%
1876   }
1877 \else
1878   \DeclareRobustCommand*\up{}%
1879   \texorpdfstring{\@ifstar{\FB@up@fake}{\textsuperscript}}{}%
1880   }
1881 \fi

```

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

```

1882 \iffBLowercaseSuperscripts
1883 \else
1884   \renewcommand*\FB@lc[1]{##1}%
1885 \fi

```

This is for `koma-script`, `memoir` and `beamer` classes. If the caption delimiter has been user customised, leave it unchanged. Otherwise, force the colon to behave properly in French (add locally `\autospace@beforeFDP` in case of `AutoSpacePunctuation=false`) and change the caption delimiter to `\CaptionSeparator` if `CustomiseFigTabCaptions` has been set to `true`.

```

1886 \iffB@koma
1887   \ifx\captionformat\FB@std@capsep
1888     \iffBCustomiseFigTabCaptions
1889       \renewcommand*\captionformat{\CaptionSeparator}%

```

```

1890     \else
1891       \renewcommand*{\captionformat}{\autospace@beforeFDP : \ }}%
1892     \fi
1893   \fi
1894 \fi
1895 \@ifclassloaded{memoir}%
1896   {\ifx\@contdelim\FB@std@capsep
1897     \ifFBCustomiseFigTabCaptions
1898       \captiondelim{\CaptionSeparator}%
1899     \else
1900       \captiondelim{\autospace@beforeFDP : }}%
1901   \fi
1902 \fi}}%
1903 \@ifclassloaded{beamer}%
1904   {\protected@edef\FB@capsep{%
1905     \csname beamer@@tmpl@caption label separator\endcsname}%
1906   \ifx\FB@capsep\FB@std@capsep
1907     \ifFBCustomiseFigTabCaptions
1908       \defbeamertemplate{caption label separator}{FBcustom}{%
1909         \CaptionSeparator}%
1910       \setbeamertemplate{caption label separator}[FBcustom]%
1911     \else
1912       \defbeamertemplate{caption label separator}{FBcolon}{%
1913         \autospace@beforeFDP : }}%
1914       \setbeamertemplate{caption label separator}[FBcolon]%
1915     \fi
1916   \fi}}%

```

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

```

1917 \ifBShowOptions
1918   \GenericWarning{* }{%
1919     *** List of possible options for babel-french ***\MessageBreak
1920     [Default values between brackets when french is loaded *LAST*]%
1921     \MessageBreak
1922     ShowOptions [false]\MessageBreak
1923     StandardLayout [false]\MessageBreak
1924     GlobalLayoutFrench [true]\MessageBreak
1925     PartNameFull [true]\MessageBreak
1926     IndentFirst [true]\MessageBreak
1927     ListItemsAsPar [false]\MessageBreak
1928     StandardListSpacing [false]\MessageBreak
1929     StandardItemizeEnv [false]\MessageBreak
1930     StandardEnumerateEnv [false]\MessageBreak
1931     StandardItemLabels [false]\MessageBreak
1932     ItemLabels=\textendash, \textbullet,
1933     \protect\ding{43},... [\textendash]\MessageBreak
1934     ItemLabeli=\textendash, \textbullet,

```

```

1935     \protect\ding{43},... [\textendash]\MessageBreak
1936 ItemLabelii=\textendash, \textbullet,
1937     \protect\ding{43},... [\textendash]\MessageBreak
1938 ItemLabeliii=\textendash, \textbullet,
1939     \protect\ding{43},... [\textendash]\MessageBreak
1940 ItemLabeliv=\textendash, \textbullet,
1941     \protect\ding{43},... [\textendash]\MessageBreak
1942 StandardLists [false]\MessageBreak
1943 ListOldLayout [false]\MessageBreak
1944 FrenchFootnotes [true]\MessageBreak
1945 AutoSpaceFootnotes [true]\MessageBreak
1946 AutoSpacePunctuation [true]\MessageBreak
1947 ThinColonSpace [false]\MessageBreak
1948 OriginalTypewriter [false]\MessageBreak
1949 UnicodeNoBreakSpaces [false]\MessageBreak
1950 og= <left quote character>, fg= <right quote character>%
1951 INGuillSpace [false]\MessageBreak
1952 EveryParGuill=open, close, none [open]\MessageBreak
1953 EveryLineGuill=open, close, none
1954         [open in LuaTeX, none otherwise]\MessageBreak
1955 InnerGuillSingle [false]\MessageBreak
1956 ThinSpaceInFrenchNumbers [false]\MessageBreak
1957 SmallCapsFigTabCaptions [true]\MessageBreak
1958 CustomiseFigTabCaptions [true]\MessageBreak
1959 OldFigTabCaptions [false]\MessageBreak
1960 FrenchSuperscripts [true]\MessageBreak
1961 LowercaseSuperscripts [true]\MessageBreak
1962 SuppressWarning [false]\MessageBreak
1963 \MessageBreak
1964 *****%
1965 \MessageBreak\protect\frenchsetup{ShowOptions}}
1966 \fi
1967 }

```

At `\begin{document}`, we have to provide an `\xspace` command in case the `xspace` package is not loaded and execute `\FBprocess@options`.

```

1968 \AtBeginDocument{%
1969   \providecommand*\xspace{\relax}%

```

Let's now process the remaining options, either not explicitly set by `\frenchsetup{}` or possibly modified by packages loaded after `babel-french`.

```

1970   \FBprocess@options

```

When option `UnicodeNoBreakSpaces` is `true` (LuaLaTeX only) we need to redefine `\FBmedkern`, `\FBthickkern` and `\FBthousandsep` as Unicode characters.

```

1971   \ifFBucsNBSP
1972     \renewcommand*\FBmedkern{\char"202F\relax}%

```

```

1973 \renewcommand*{\FBthickkern}{\char"A0\relax}%
1974 \ifFBThinSpaceInFrenchNumbers
1975 \renewcommand*{\FBthousandsep}{\char"202F\relax}%
1976 \else
1977 \renewcommand*{\FBthousandsep}{\char"A0\relax}%
1978 \fi
1979 \fi

```

Finally, with pdfLaTeX, when OT1 encoding is in use at the `\begin{document}` a warning is issued; `\encodingdefault` being defined as ‘long’, the test would fail if `\FBOTone` was defined with `\newcommand*`!

```

1980 \begingroup
1981 \newcommand{\FBOTone}{OT1}%
1982 \ifx\encodingdefault\FBOTone
1983 \FBWarning{OT1 encoding should not be used for French.%
1984 \MessageBreak
1985 Add \protect\usepackage[T1]{fontenc} to the
1986 preamble\MessageBreak of your document; reported}%
1987 \fi
1988 \endgroup
1989 }

```

2.12 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.

```

1990 \let\listORI\list
1991 \let\endlistORI\endlist
1992 \newdimen\FB@parskip
1993 \def\FB@listVsettings{%
1994 \setlength{\topsep}{0.8ex plus 0.4ex minus 0.4ex}%
1995 \setlength{\partopsep}{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@parskip.`

```
1996 \FB@parskip=\parskip
1997 \addtolength{\topsep}{-\FB@parskip}%
1998 \addtolength{\partopsep}{\FB@parskip}%
1999 \setlength{\itemsep}{0.4ex plus 0.2ex minus 0.2ex}%
2000 \setlength{\parsep}{0.4ex plus 0.2ex minus 0.2ex}%
```

(v3.5q) 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.

```
2001 \ifdim\FB@parskip>0pt
2002 \setlength{\parsep}{\FB@parskip}%
2003 \addtolength{\itemsep}{-\FB@parskip}%
2004 \fi
2005 }
2006 \def\listFB#1#2{\listORI{#1}{\FB@listVsettings #2}}
2007 \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 2.11).
- 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 2008 \renewcommand*{\FrenchLabelItem}{\textemdash}
\Frlabelitemiii 2009 \renewcommand*{\Frlabelitemi}{\FrenchLabelItem}
\Frlabelitemiv 2010 \renewcommand*{\Frlabelitemii}{\FrenchLabelItem}
2011 \renewcommand*{\Frlabelitemiii}{\FrenchLabelItem}
2012 \renewcommand*{\Frlabelitemiv}{\FrenchLabelItem}
```

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

```
2013 \newdimen\listindentFB
2014 \setlength{\listindentFB}{-1pt}
2015 \newdimen\descindentFB
```

```

2016 \setlength{\descindentFB}{-1pt}
2017 \newdimen\labelindentFB
2018 \setlength{\labelindentFB}{-1pt}
2019 \newdimen\labelwidthFB
2020 \setlength{\labelwidthFB}{-1pt}

```

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

```

2021 \newdimen\leftmarginFB
2022 \def\FB@listHsettings{%
2023   \ifFBListItemsAsPar

```

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

```

2024   \itemindent=\labelindentFB
2025   \advance\itemindent by \labelwidthFB
2026   \advance\itemindent by \labelsep
2027   \leftmargini\z@
2028   \bbl@for\FB@dp {2, 3, 4, 5, 6}%
2029     {\csname leftmargin\romannumeral\FB@dp\endcsname =
2030       \labelindentFB}%
2031   \else

```

Default layout: labels hanging into the list left margin.

```

2032   \leftmarginFB=\labelwidthFB
2033   \advance\leftmarginFB by \labelsep
2034   \bbl@for\FB@dp {1, 2, 3, 4, 5, 6}%
2035     {\csname leftmargin\romannumeral\FB@dp\endcsname =
2036       \leftmarginFB}%
2037   \advance\leftmargini by \listindentFB

```

(v3.5q) Same 'parindent' for paragraphs in lists' items (was null as in standard lists).

```

2038   \listparindent=\parindent
2039   \fi
2040   \leftmargin=\csname leftmargin%
2041     \ifnum\@listdepth=\@ne i\else ii\fi\endcsname
2042 }

```

`\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` unless option `ListOldLayout` is `true` (compatibility with lists up to v2.5k).

```

2043 \def\FB@itemizesettings{%
2044   \ifFBStandardListSpacing
2045   \else
2046     \setlength{\topsep}{\z@}%
2047     \setlength{\partopsep}{\z@}%

```

```

2048 \FB@parskip=\parskip
2049 \addtolength{\topsep}{-\FB@parskip}%
2050 \addtolength{\partopsep}{\FB@parskip}%
2051 \setlength{\itemsep}{\z@}%
2052 \setlength{\parsep}{\z@}%
2053 \ifdim\FB@parskip>0pt
2054 \setlength{\parsep}{\FB@parskip}%
2055 \addtolength{\itemsep}{-\FB@parskip}%
2056 \fi
2057 \fi
2058 \settowidth{\labelwidth}{\csname\@itemitem\endcsname}%
2059 \ifFBListOldLayout
2060 \setlength{\leftmargin}{\labelwidth}%
2061 \addtolength{\leftmargin}{\labelsep}%
2062 \addtolength{\leftmargin}{\parindent}%
2063 \else
2064 \FB@listHsettings
2065 \fi
2066 }

```

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

```

2067 \def\itemizeFB{%
2068 \ifnum \@itemdepth >\thr@@\@toodeep\else
2069 \advance\@itemdepth by \@ne
2070 \edef\@itemitem{labelitem\romannumeral\the\@itemdepth}%
2071 \expandafter
2072 \listORI
2073 \csname\@itemitem\endcsname
2074 \FB@itemizesettings
2075 \fi
2076 }
2077 \let\enditemizeFB\endlistORI

2078 \def\setlabelitemsFB{%
2079 \let\labelitemi\Frlabelitemi
2080 \let\labelitemii\Frlabelitemii
2081 \let\labelitemiii\Frlabelitemiii
2082 \let\labelitemiv\Frlabelitemiv
2083 \ifdim\labelwidthFB<\z@
2084 \settowidth{\labelwidthFB}{\FrenchLabelItem}%
2085 \fi
2086 }
2087 \def\setlistindentFB{%
2088 \ifdim\labelindentFB<\z@
2089 \ifdim\parindent=\z@
2090 \setlength{\labelindentFB}{1.5em}%

```



```

2091   \else
2092     \setlength{\labelindentFB}{\parindent}%
2093   \fi
2094 \fi
2095 \ifdim\listindentFB<\z@
2096   \ifdim\parindent=\z@
2097     \setlength{\listindentFB}{1.5em}%
2098   \else
2099     \setlength{\listindentFB}{\parindent}%
2100   \fi
2101 \fi
2102 \ifdim\descindentFB<\z@
2103   \ifFBListItemsAsPar
2104     \setlength{\descindentFB}{\labelindentFB}%
2105   \else
2106     \setlength{\descindentFB}{\listindentFB}%
2107   \fi
2108 \fi
2109 }

```

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

```

2110 \def\enumerateFB{%
2111   \ifnum \@enumdepth >\thr@@\@toodeep\else
2112     \advance\@enumdepth by \@ne
2113     \edef\@enumctr{enum\romannumeral\the\@enumdepth}%
2114     \expandafter
2115     \list
2116       \csname label\@enumctr\endcsname
2117       {\FB@listHsettings
2118         \usecounter\@enumctr\def\makelabel##1{\hss\llap{##1}}}%
2119   \fi
2120 }
2121 \let\endenumerateFB\endlistORI

```

`\descriptionFB` Same tuning for the `description` environment (see `classes.dtx` for the original definition). Customisable dimen `\descindentFB`, which defaults to `\listindentFB`, is added to `\itemindent` (first level only). When `\descindentFB=0pt` (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.

```

2122 \def\descriptionFB{%

```

```

2123 \list}{\FB@listHsettings
2124 \labelwidth=\z@
2125 \ifFBListItemsAsPar
2126 \itemindent=\descindentFB
2127 \else
2128 \itemindent=-\leftmargin
2129 \ifnum\@listdepth=\@ne
2130 \ifdim\descindentFB=\z@
2131 \ifdim\listindentFB>\z@
2132 \leftmargini=\listindentFB
2133 \leftmargin=\leftmargini
2134 \itemindent=-\leftmargin
2135 \fi
2136 \else
2137 \advance\itemindent by \descindentFB
2138 \fi
2139 \fi
2140 \fi
2141 \let\makelabel\descriptionlabel}%
2142 }
2143 \let\enddescriptionFB\endlistORI

```

`\update@frenchlists` `\update@frenchlists` will set up lists according to the final options (default or part of `\frenchsetup`) eventually overruled in `\FBprocess@options`).

```

2144 \def\update@frenchlists{%
2145 \setlistindentFB
2146 \ifFBStandardListSpacing
2147 \else \let\list\listFB \fi
2148 \ifFBStandardItemizeEnv
2149 \else \let\itemize\itemizeFB \fi
2150 \ifFBStandardItemLabels
2151 \else \setlabelitemsFB \fi
2152 \ifFBStandardEnumerateEnv
2153 \else \let\enumerate\enumerateFB \let\description\descriptionFB \fi
2154 }

```

If `GlobalLayoutFrench=true`, nothing has to be done at language's switches regarding lists. Otherwise, `\extrasfrench` saves the standard settings for lists and then executes `\update@frenchlists`. In both cases, there is nothing to do for lists in `\noextrasfrench`.

In order to ensure compatibility with packages customising lists, the command `\update@frenchlists` should not be included in the first call to `\extrasfrench` which occurs *before* the relevant flags are finally set, so we define `\FB@uf1` as `\relax`, it will be redefined later 'AtBeginDocument' by `\FBprocess@options` as `\update@frenchlists`, see p. 70.

Lists' layout changes at language switches only if `GlobalLayoutFrench=false`.

```

2155 \def\FB@ufl{\relax}
2156 \def\bbl@frenchlistlayout{%
2157   \ifFBGlobalLayoutFrench
2158   \else
2159     \babel@save\list          \babel@save\itemize
2160     \babel@save\enumerate    \babel@save\description
2161     \babel@save\labelitemi   \babel@save\labelitemii
2162     \babel@save\labelitemiii \babel@save\labelitemiv
2163     \FB@ufl
2164   \fi
2165 }
2166 \addto\extrasfrench{\bbl@frenchlistlayout}

```

2.13 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 two cases:

- a) `GlobalLayoutFrench=false`,
- b) `IndentFirst=true` and French isn't the main language.

```

2167 \def\bbl@frenchindent{%
2168   \ifFBGlobalLayoutFrench\else\babel@save\@afterindentfalse\fi
2169   \ifFBIndentFirst
2170     \ifFB@mainlanguage@FR\else\babel@save\@afterindentfalse\fi
2171     \let\@afterindentfalse\@afterindenttrue
2172     \@afterindenttrue
2173   \fi}
2174 \addto\extrasfrench{\bbl@frenchindent}

```

2.14 Formatting footnotes

The layout of footnotes is controlled by two flags `\ifFBAutoSpaceFootnotes` and `\ifFBFrenchFootnotes` which are set by options of `\frenchsetup` (see section 2.11). 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!).

`\@makefntextFB` We then 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).

The value of `\parindentFFN` will be redefined at the `\begin{document}`, as the maximum of `\parindent` and `1.5em` *unless* it has been set in the preamble (the weird value `10in` is just for testing whether `\parindentFFN` has been set or not).

```
2175 \newdimen\parindentFFN
2176 \parindentFFN=10in
```

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

```
2177 \newcommand*{\dotFFN}{.}
2178 \newcommand*{\kernFFN}{\kern .5em}
2179 \newdimen\FBfnindent
```

`\@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.

```
2180 \iffB@koma
2181 \let\@makefntextORI\@makefntext
2182 \let\@@makefnmarkORI\@@makefnmark
```

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

```
2183 \deffootnote[\FBfnindent]{0pt}{\parindentFFN}%
2184           {\thefootnotemark\dotFFN\kernFFN}
2185 \let\@makefntextFB\@makefntext
2186 \let\@@makefnmarkFB\@@makefnmark
```

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

```
2187 \deffootnote[\parindentFFN]{0pt}{\parindentFFN}%
2188           {\textsuperscript{\thefootnotemark}}
2189 \let\@makefntextTH\@makefntext
2190 \let\@@makefnmarkTH\@@makefnmark
```

Restore the original definitions.

```
2191 \let\@makefntext\@makefntextORI
2192 \let\@@makefnmark\@@makefnmarkORI
2193 \fi
```

Definitions for the `memoir` class:

```
2194 \@ifclassloaded{memoir}
```

(see original definition in `memman.pdf`)

```
2195 {\newcommand{\@makefntextFB}[1]{%
2196   \def\footscript##1{##1\dotFFN\kernFFN}%
2197   \setlength{\footmarkwidth}{\FBfnindent}%
2198   \setlength{\footmarksep}{-\footmarkwidth}%
2199   \setlength{\footparindent}{\parindentFFN}%
```

```

2200     \makefootmark #1}%
2201   }{}

```

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.

```

2202 \@ifclassloaded{beamer}
2203   {\def\@makefntextFB#1{%
2204     \def\insertfootnotetext{#1}%
2205     \def\insertfootnotemark{\insertfootnotemarkFB}%
2206     \usebeamertemplate***{footnote}}%
2207   \def\insertfootnotemarkFB{%
2208     \usebeamercolor[fg]{footnote mark}%
2209     \usebeamerfont*{footnote mark}%
2210     \llap{\@thefnmark}\dotFFN\kernFFN}%
2211   }{}

```

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)!

```

2212 \providecommand*\insertfootnotemarkFB{%
2213   \parindent=\parindentFFN
2214   \rule\z@\footnotesep
2215   \setbox\@tempboxa\hbox{\@thefnmark}%
2216   \ifdim\wd\@tempboxa>\z@
2217     \llap{\@thefnmark}\dotFFN\kernFFN
2218   \fi}
2219 \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 the `\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: let’s define a customisable thin space which will be added before footnote’s call by `\@footnotemarkFB`.

```

2220 \newcommand*\FBfnmarkspace{\kern .5\fontdimen2\font}
2221 \def\@footnotemarkFB{\leavevmode\unskip\unkern
2222     \protect\FBfnmarkspace\@footnotemarkORI}%

```

Switching between French or Standard layout for footnotes is done ‘AtBeginDocument’. The LuaTeX command `\localleftbox` and `\FBeverypar@quote` used by `\frquote{}` have to be reset inside footnotes; done for LaTeX based formats only.

```
2223 \providecommand\localleftbox[1]{}
2224 \AtBeginDocument{%
```

```
2224 \AtBeginDocument{%
```

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

```
2225 \@ifpackageloaded{footnotebackref}%
2226     {\FBFrenchFootnotesfalse
2227     \PackageWarning{french.1df}%
2228         {footnotebackref package loaded.\MessageBreak
2229         babel-french will NOT customise footnotes;%
2230         \MessageBreak reported}}%
2231     {}%
```

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.

```
2232 \@ifpackageloaded{bigfoot}%
2233     {\PackageWarning{french.1df}%
2234     {bigfoot package in use.\MessageBreak
2235     babel-french will NOT customise footnotes;%
2236     \MessageBreak reported}}%
```

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

```
2237     {\let\@footnotemarkORI\@footnotemark
2238     \ifFBAutoSpaceFootnotes
2239     \let\@footnotemark\@footnotemarkFB
2240     \fi
2241     \ifdim\parindentFFN<10in
2242     \else
2243     \parindentFFN=\parindent
2244     \ifdim\parindentFFN<1.5em \parindentFFN=1.5em \fi
2245     \fi
2246     \settowidth{\FBfnindent}{\dotFFN\kernFFN}%
2247     \addtolength{\FBfnindent}{\parindentFFN}%
2248     \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).

```
2249     \ifFB@koma
2250     \let\@@makefnmarkORI\@@makefnmark
2251     \long\def\@makefntext#1{%
```

```

2252     \lcalleftbox{}}%
2253     \let\FBeverypar@save\FBeverypar@quote
2254     \let\FBeverypar@quote\relax
2255     \ifFBFrenchFootnotes
2256         \ifx\footnote\thanks
2257             \let\@@makefnmark\@@makefnmarkTH
2258             \@makefntextTH{#1}
2259         \else
2260             \let\@@makefnmark\@@makefnmarkFB
2261             \@makefntextFB{#1}
2262         \fi
2263     \else
2264         \let\@@makefnmark\@@makefnmarkORI
2265         \@makefntextORI{#1}%
2266     \fi
2267     \let\FBeverypar@quote\FBeverypar@save
2268     \lcalleftbox{\FBeveryline@quote}}%
2269 \else

```

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

```

2270     \ifclassloaded{memoir}%
2271     {
2272         \setlength{\thanksmarkwidth}{\parindentFFN}%
2273         \setlength{\thanksmarksep}{-\thanksmarkwidth}%
2274     \fi
2275     }{}%

```

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

```

2276     \ifclassloaded{beamer}%
2277     {
2278         \ifdim\parindentFFN=1.5em\else
2279             \FBWarning{%
2280                 \protect\parindentFFN\space is ineffective%
2281                 \MessageBreak within the beamer class.%
2282                 \MessageBreak Reported}%
2283         \fi
2284     \fi
2285     }{}%

```

Definition of \@makefntext for all other classes:

```

2286     \long\def\@makefntext#1{%
2287         \lcalleftbox{}}%
2288     \let\FBeverypar@save\FBeverypar@quote
2289     \let\FBeverypar@quote\relax
2290     \ifFBFrenchFootnotes
2291         \@makefntextFB{#1}%

```

```

2292         \else
2293             \@makefntextORI{#1}%
2294         \fi
2295         \let\FBeverypar@quote\FBeverypar@save
2296         \localleftbox{\FBeveryline@quote}}%
2297     \fi
2298 }%
2299 }

```

For compatibility reasons, we provide definitions for the commands dealing with the layout of footnotes in `babel-french` version 1.6. `\frenchsetup{}` (see in section 2.11) should be preferred for setting these options. `\StandardFootnotes` may still be used locally (in minipages for instance), that's why the test `\ifFBFrenchFootnotes` is done inside `\@makefntext`.

```

2300 \newcommand*{\AddThinSpaceBeforeFootnotes}{\FBAutoSpaceFootnotestruer}
2301 \newcommand*{\FrenchFootnotes}{\FBFrenchFootnotestruer}
2302 \newcommand*{\StandardFootnotes}{\FBFrenchFootnotesfalse}

```

2.15 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.

```

2303 \FBclean@on@exit
2304 \ldf@finish\CurrentOption
2305 \let\loadlocalcfg\FB@llc
2306 </french>

```

2.16 Files `frenchb.ldf`, `francais.ldf`, `canadien.ldf` and `acadian.ldf`

Babel now expects a `<lang>.ldf` file for each `<lang>`. So we create portmanteau `.ldf` files for options `canadien`, `francais`, `frenchb` and `acadian`. These files themselves only load `french.ldf` which does the real work. Warn users about options `canadien`, `frenchb` and `francais` being deprecated and force recommended options `acadian` or `french`.

```

2307 <*acadian>
2308 \PackageInfo{acadian.ldf}%
2309 {`acadian' dialect is currently\MessageBreak
2310  *absolutely identical* to the\MessageBreak
2311  `french' language; reported}
2312 </acadian>
2313 <*canadien>
2314 \PackageWarning{canadien.ldf}%
2315 {Option `canadien' for Babel is *deprecated*,\MessageBreak

```



```

2316   it might be removed sooner or later. Please\MessageBreak
2317   use `acadian' instead; reported}%
2318 \def\CurrentOption{acadian}

2319 \def\datecanadien{\dateacadian}
2320 \def\captionscanadien{\captionscadian}
2321 \def\extrascanadien{\extrasacadian}
2322 \def\noextrascanadien{\noextrasacadian}
2323 </canadien>
2324 <*français>
2325 \PackageWarning{français.1df}%
2326   {Option `français' for Babel is *deprecated*,\MessageBreak
2327   it might be removed sooner or later. Please\MessageBreak
2328   use `french' instead; reported}%
2329 \chardef\l@français\l@french
2330 \def\CurrentOption{french}
2331 </français>

```

Compatibility code for Babel pre-3.13: frenchb.1df could be loaded with options `acadian`, `canadien`, `frenchb` or `français`.

```

2332 <*frenchb>
2333 \def\bb1@tempa{frenchb}
2334 \ifx\CurrentOption\bb1@tempa
2335   \chardef\l@frenchb\l@french
2336   \def\CurrentOption{french}
2337   \PackageWarning{babel-french}%
2338     {Option `frenchb' for Babel is *deprecated*,\MessageBreak
2339     it might be removed sooner or later. Please\MessageBreak
2340     use `french' instead; reported}
2341 \else
2342   \def\bb1@tempa{français}
2343   \ifx\CurrentOption\bb1@tempa
2344     \chardef\l@français\l@french
2345     \def\CurrentOption{french}

```

Plain formats: no warning when `français.sty` loads `frenchb.1df` (Babel pre-3.13).

```

2346   \ifx\magnification\undefined
2347     \PackageWarning{babel-french}%
2348       {Option `français' for Babel is *deprecated*,\MessageBreak
2349       it might be removed sooner or later. Please\MessageBreak
2350       use `french' instead; reported}
2351   \fi
2352 \else
2353   \def\bb1@tempa{canadien}
2354   \ifx\CurrentOption\bb1@tempa
2355     \def\CurrentOption{acadian}
2356     \PackageWarning{babel-french}%
2357       {Option `canadien' for Babel is *deprecated*,\MessageBreak

```

2358 it might be removed sooner or later. Please\MessageBreak
2359 use `acadian' instead; reported}
2360 \fi
2361 \fi
2362 \fi
2363 </frenchb>
2364 <acadian/canadien/frenchb/francais>\input french.1df\relax
2365 <acadian/canadien>\let\extrasacadian\extrasfrench
2366 <acadian/canadien>\let\noextrasacadian\noextrasfrench
2367 <acadian/canadien/frenchb/francais/french>\endinput

3 Change History

Changes listed in reverse order (latest first) and not older than v3.3 (2018).

v3.6b	<code>\NoAutoSpacing</code> : <code>\NoAutoSpacing</code> must be inhibited in bookmarks. 39	<code>\FB@xetex@punct@french</code> : <code>\shorthandon</code> and <code>\shorthandoff</code> are no longer redefined (it broke <code>\shorthandoff*</code>). 34
v3.6a	General: Internal ‘l3keys’ replaces package ‘keyval’ for options’ management. 59	<code>frenchb.lua</code> : Opening guill.: look ahead when next is a penalty (nobreak space). 29
	<code>\@footnotemark</code> : Allow customisation of the space added in <code>\footnotemarkFB</code> 82	v3.5n
	<code>\degres</code> : Simplify <code>\degres</code> definition for text and math mode: <code>\textdegree</code> always defined (TS1) since 2019. 49	General: <code>\FBGlobalLayoutFrench</code> no longer set to false when French is not the main language. 60
v3.5s	General: Footnotes: no customising of <code>\@footnotetext</code> when the <code>footnotebackref</code> package is loaded. Just warn the user. 83	<code>\bbl@frenchindent</code> : <code>\bbl@frenchindent</code> changed. <code>\bbl@nonfrenchindent</code> removed. 80
	<code>frenchb.lua</code> : A ‘.’ followed by ‘-’ or a ligature should not trigger spacing. 27	<code>\bsc</code> : Added command <code>\bname</code> (no small caps). 48
v3.5r	General: Compatibility with <code>ucharclasses</code> package added. . . 33	v3.5m
v3.5q	<code>\listFB</code> : Bug correction: <code>\parsep</code> should be related to <code>\parskip</code> and <code>\listparindent</code> to <code>\parindent</code> . . 74	<code>\FBtextellipsis</code> : No longer redefine <code>\dots</code> , only <code>\textellipsis</code> ’s default definition is changed in French. 57
v3.5p	<code>\DecimalMathComma</code> : <code>\DecimalMathComma</code> can again be used in the preamble for a global action. It now works as expected inside a group. 49	v3.5l
	<code>\frquote</code> : <code>\FBeveryline@quote</code> : no need for a penalty inside a <code>\localleftbox</code> 43	General: No warning about <code>\@makecaption</code> for more classes. . 56
v3.5o	General: <code>\shorthandon</code> and <code>\shorthandoff</code> are no longer redefined in LuaTeX (it broke <code>\shorthandoff*</code>). 32	<code>\captionsfrench</code> : Redefine <code>\fnum@figure</code> and <code>\fnum@table</code> separately. 53
		v3.5k
		General: <code>\degre</code> , <code>\degres</code> , <code>\circflexe</code> , <code>\tild</code> , <code>\boi</code> and <code>\at</code> are now safe in bookmarks. . 48
		<code>\pdfstringdefDisableCommands</code> dropped. 73
		Reorganise warnings about ‘.’ in captions, according to enhancements in <code>caption.sty</code> v3.5a. 56
		<code>\bsc</code> : <code>\bsc</code> now relies on <code>\texorpdfstring</code> to be safe in bookmarks. 48
		<code>\captionsfrench</code> : Small caps removed in <code>\figurename</code> and

<ul style="list-style-type: none"> <ul style="list-style-type: none"> <code>\tablename</code>, use <code>\fnum@figure</code> and <code>\fnum@table</code> instead. 53 <code>\FB@fg</code>: <code>\FB@og</code> and <code>\FB@fg</code> now rely on <code>\texorpdfstring</code> to be safe in bookmarks. 40 <code>\frquote</code>: <code>\frquote</code> now relies on <code>\texorpdfstring</code> to be safe in bookmarks. 42 <code>\fup</code>: <code>\up</code> and <code>\fup</code> now rely on <code>\texorpdfstring</code> to be safe in bookmarks. 45 <code>\no</code>: <code>\no</code>, <code>\nos</code>, <code>\No</code>, <code>\Nos</code>, <code>\primo</code>, <code>\fprimo</code>, now rely on <code>\texorpdfstring</code> to be safe in bookmarks. 47 	<ul style="list-style-type: none"> allowed again only for backward compatibility (deprecated). 86
<ul style="list-style-type: none"> v3.5j <ul style="list-style-type: none"> General: For memoir, koma-script and beamer captions, <code>\FB@std@sep</code> has to be defined before activating the colon. 36 	<ul style="list-style-type: none"> v3.5e <ul style="list-style-type: none"> General: <code>StandardLayout</code> and <code>GlobalLayoutFrench</code> options can no longer be toggled when French is not the main language. 60 <code>\frquote</code>: Make resettings global on exit. 44 new command <code>\NoEveryParQuote</code>. 44 reset <code>\FB@addGUIspace</code> attribute inside <code>\localleftbox</code> (LuaTeX). . . 43
<ul style="list-style-type: none"> v3.5i <ul style="list-style-type: none"> <code>\frenchsetup</code>: For memoir, koma-script and beamer classes, leave caption delimiter unchanged if it has been user customised. . . 71 	<ul style="list-style-type: none"> v3.5d <ul style="list-style-type: none"> <code>\frenchsetup</code>: <code>ReduceListSpacing</code> option deprecated: see <code>StandardListSpacing</code>. 62
<ul style="list-style-type: none"> v3.5h <ul style="list-style-type: none"> <code>frenchb.lua</code>: 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. 26 Code reorganised for better efficiency. 26 	<ul style="list-style-type: none"> v3.5c <ul style="list-style-type: none"> General: Remove grouping inside <code>\@makefntext</code>, <code>\localleftbox</code> and <code>\FBeverypar@quote</code> saved and restored instead. 83 <code>\frquote</code>: <code>\FBeverypar@quote</code>'s value now properly reset across level changes. 43 <code>\noextrasfrench</code>: <code>\lccode</code> of quote 0x27 changed from 0x2019 to 0x27 for Unicode engines. 17
<ul style="list-style-type: none"> v3.5g <ul style="list-style-type: none"> <code>frenchb.lua</code>: The kerning callback is a bit specific: adding code with <code>add_to_callback</code> actually deletes the legacy kerning as pointed out by Marcel Krüger on SE. 26 	<ul style="list-style-type: none"> v3.5b <ul style="list-style-type: none"> General: Reset <code>\FBeverypar@quote</code> locally inside <code>\@makefntext</code>. Needed by <code>\frquote</code>. 83 <code>\frquote</code>: New command <code>\FB@addquote@everypar</code> to manage <code>\everypar</code>: <code>\frquote</code> failed when used immediately after a sectionning command. . . 42
<ul style="list-style-type: none"> v3.5f <ul style="list-style-type: none"> General: <code>\l@canadien</code> was defined too early in file 'canadien.ldf': <code>\l@acadian</code> might not be defined. 15 <code>\selectlanguage{canadien}</code> 	<ul style="list-style-type: none"> v3.5a <ul style="list-style-type: none"> General: New optional layout for lists: lists' items can be typeset as paragraphs with indented labels while the default leaves the labels hanging into the left margin. . . . 76

<code>\descriptionFB: ListItemsAsPar</code> option taken into account for description lists.	78	finally removed (babel bug reported).	44
<code>\frenchsetup: New option</code> <code>ListItemsAsPar</code> for displaying lists' items "as paragraphs".	62	<code>\extrasfrench: Change</code> <code>\(no)extras\CurrentOption</code> to <code>\(no)extrasfrench</code> . <code>\(no)extrasacadian</code> will be defined as <code>\(no)extrasfrench</code> in file <code>acadian.ldf</code>	16
v3.4d <code>\frenchsetup: New test for deciding</code> about utf8 encoding for keys <code>og</code> and <code>fg</code> (the former one fails with LaTeX 2018 release).	66	<code>frenchb.lua: Global 'FBsp' table</code> added; local function ' <code>get_glue</code> ' changed into global ' <code>FBget_glue</code> '.	24
v3.4c <code>\ifFBXeTeX: Reverting to former test,</code> beware of <code>\XeTeXrevision</code> left as <code>\relax</code> by careless testing.	16	v3.3d <code>frenchb.lua: In default mode, for ':'</code> only, check if next node is a glyph or not. If it is, turn the ' <code>auto</code> ' flag to false (avoids spurious spaces in URLs, MSDOS paths or 10:35).	27
v3.4b <code>\datefrench: Do not redefine \date</code> as <code>\frenchdate</code> in French.	44	v3.3c General: LaTeX 2017-04-15 defines TU encoding for Unicode engines, <code>fontspec</code> is no longer required.	74
v3.4a General: <code>\LdfInit</code> checks <code>\FBclean@on@exit</code> instead of <code>\captionsfrench</code> (undefined in PLain). Prevents loading <code>french.ldf</code> again with <code>acadian</code> option.	14	New command <code>\FBthousandsep</code> to customise numprint.	52
<code>babel-french</code> now requires eTeX.	14	New configurable kerns <code>\FBmedkern</code> , and <code>\FBthickkern</code> suitable for HTML translation.	47
Lua function <code>token.get_meaning</code> requires LuaTeX 1.0.	22	Reorganise warnings when the caption, subcaption or floatrow packages are loaded before <code>babel/french</code>	56
New <code>\FBgspchar</code> to customise the space character to be used for <code>\og</code> and <code>\fg</code> with the <code>UnicodeNoBreakSpaces</code> option.	40	Reset <code>\localleftbox</code> locally inside <code>\@makefnstext</code> . Needed by <code>\frquote</code> with LuaTeX.	83
New attribute <code>\FB@dialect</code> for the French dialect <code>acadian</code>	21	<code>\frenchsetup: New option</code> ' <code>UnicodeNoBreakSpaces</code> ' for html translators (LuaLaTeX only).	65
New command <code>\FBsetspaces</code> to fine tune spacing independently in French and in French dialects.	18	<code>frenchb.lua: Function 'get_glue'</code> robustified. ' <code>french_punctuation</code> ' can insert Unicode characters instead of glues.	23
Patch for koma-script classes moved here, after <code>\ifFBPartNameFull</code> is defined, so that it applies to <code>\extrasacadian</code> too: <code>\AtEndOfPackage</code> is too late.	60	v3.3b General: Generate portmanteau files <code>acadian.ldf</code> , <code>canadien.ldf</code> , <code>frenchb.ldf</code> , and <code>français.ldf</code> and warn about deprecated options.	85
Shrink/stretch removed in <code>\FBthousandsep</code>	52		
Toks <code>\FBcolonsp</code> , <code>\FBthinsp</code> and <code>\FBguillsp</code> removed.	18		
<code>\datefrench: Specific code for Plain</code>			

New ‘if’ <code>\ifFBfrench</code> to replace <code>\iflanguage</code> test which is based on patterns.	16		
v3.3a			
General: Compatibility code for pre 2015/10/01 LaTeX release removed, see <code>ltnews23.tex</code>	21		
Skip <code>\FBguillskip</code> for LuaTeX replaced by toks <code>\FBguillsp</code>	18		
<code>\captionsfrench</code> : Commands <code>\frenchpartfirst</code> , <code>\frenchpartsecond</code> and			
			<code>\frenchpartnameord</code> added.
			53
		<code>\FBthinspace</code> : Skips <code>\FBcolonskip</code> and <code>\FBthinspace</code> replaced by toks <code>\FBcolonsp</code> and <code>\FBthinsp</code>	18
		<code>\frenchsetup</code> : <code>\frenchbsetup</code> is now an alias for <code>\frenchsetup</code>	62
		Options <code>INGuillSpace</code> , <code>ThinColonSpace</code> no longer delayed <code>AtBeginDocument</code>	62
		<code>\frquote</code> : <code>\FB@quotespace</code> (kern), changed into <code>\FB@guillspace</code>	42