A Babel language definition file for French

frenchb.dtx v3.5r, 2023-12-19

Daniel Flipo
daniel.flipo@free.fr

Contents

1 The French language

The file `frenchb.dtx`[1], defines all the language definition macros for the French language.
Customisation for the French language is achieved following the book “Lexique des règles typographiques en usage à l’Imprimerie Nationale’’ troisième édition (1994), ISBN-2-11-081075-0.
First version released: 1.1 (May 1996) as part of Babel-3.6beta. Version 2.0a was released in February 2007 and version 3.0a in February 2014.
`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 v3.5r are listed in subsection 1.4 p. 11.
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. 4);
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. 9.

[1] The file described in this section has version number v3.5r and was last revised on 2023-12-19.
[2] *Always* use `french` as option name for the French language, former aliases `frenchb` or `francais` are *depreciated*; expect them to be removed sooner or later!
[3] For 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. 4).

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

1. French hyphenation patterns are made active;
2. 'high punctuation' characters (`: ; ! ?`) automatically add correct spacing [5] 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. 9.
5. the space after `\dots` is removed in French.

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

1. French quotation marks can be entered using the command `\frquote{}`: `\frquote{some text}` will output « some text ». Former commands `\og` and `\fg` are kept for backward compatibility: `\og some text\fg{}` is an alternative to `\frquote{some text}`.

   If French quote characters are available on your keyboard, you can use them, to get proper spacing in LaTeX2e see option `og=«, fg=»` p. 7.

   For quotations spreading over more than one paragraph, `\frquote` will add at the beginning of every paragraph of the quotation either an opening French guillemet («), or a closing one (») or nothing depending on option `EveryParGuill=open` or `=close` or `=none`, see p. 8. 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`.
   - 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'.

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

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

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

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

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

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

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

7. Two commands 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 TEXbook p. 134). The command `\DecimalMathComma` makes the comma behave as an ordinary character *when the current language is French* (no space added); as a counterpart, if `\DecimalMathComma` is active, an explicit thin space has to be added in lists and intervals: `$(x,\,y)$`, `$[0,\,1]$`. `\StandardMathComma` switches back to the standard behaviour of the comma in French.

   The `icomma` package is an alternative workaround.

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

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

1.2 Customisation

Customisation of `babel-french` relies on command `\frenchsetup{}` (formerly called `\frenchbsetup{}`, the latter name will be kept for ever to ensure backwards compatibility), options are entered using the `keyval` 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 keyval syntax, boolean options (as ShowOptions) can be entered as ShowOptions=true or just ShowOptions, the =true part can be omitted.

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

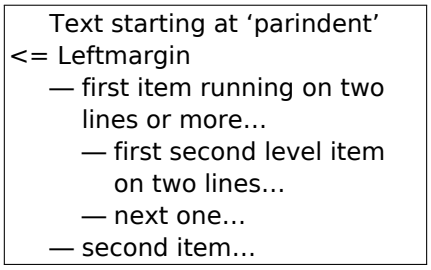
StandardLayout=true (false*) forces babel-french not to interfere with the layout: no action on any kind of lists, first paragraphs of sections are not indented (as in English), no action on footnotes; it useless unless French is the main language. This option can be used to avoid conflicts with classes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

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).

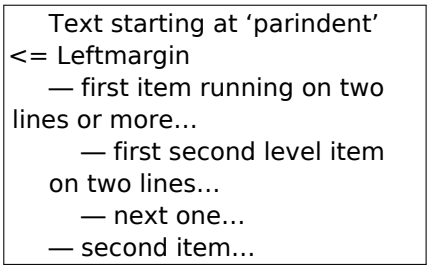
IndentFirst=false (true*); set this option to false if you do not want babel-french to force indentation of the first paragraph of sections. When French is the main language, this option applies to all languages.

PartNameFull=false (true); when true, 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:



Default French layout



With ListItemsAsPar=true

`StandardListSpacing=true (false*)`[6]; babel-french 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.

`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. Try the option `StandardLists` 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).

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

`AutoSpaceFootnotes=false (true*)`; by default `babel-french` adds a 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).

`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` (defauts to a thin space) before ‘`;`’ ‘`!`’ ‘`?`’ or `\FBcolonspace` (defauts 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 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 `lwarp` (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,

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

[8] Actually, 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…

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

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.

`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 parapraph 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` (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 parapraph 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 (`\,`).

`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 `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.

[10] Actually without stretch nor shrink.

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

`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}` to get the expected layout. 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* preceed 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); toogle 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 TEX 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 3.5?

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 sectionning 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` et `beamer` classes: redefintions 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` (formely 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 convertion into HTML of non-breaking spaces added by babel-french. Usage of lwarp (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. 7.
According to current Babel's standards, every dialect should have it's own .ldf file; starting with version 3.3b, the main support for French is in french.ldf, portmanteau files frenchb.ldf,francais.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.

What's new in version 3.2?

Version 3.2g changes the default behaviour of \frquote{} with LuaTeX based engines, the output is now the same with all engines; to recover the former behaviour, add option EveryLineGuill=open.
The handling of footnotes has been redesigned for the beamer, memoir and koma-script classes. The layout of footnotes "à la française'' should be unchanged but footnotes' customisations offered by these classes (i.e. font or color changes) are now available even when option FrenchFootnotes is true.
A long standing bug regarding the xspace package has been fixed: \xspace has been moved up from the internal command \FB@fg to \fg; \frquote{} now works properly when the xspace package is loaded.
Version 3.2b is the first one designed to work with LuaTeX v. 0.95 as included in TeXLive 2016 (LuaTeX's new glue node structure is not compatible with previous versions).
Warning to Lua(La)TeX users: starting with version 3.2b the lua code included in frenchb.lua will *not work* on older installations (TL2015 f.i.), so babel-french reverts to active characters while handling high punctuation with LuaTeX engines older than 0.95! The best way to go is to upgrade to TL2016 or equivalent asap.

Xe(La)TeX and pdf(La)TeX users can safely use babel-french v. 3.2b and later on older installations too.
The internals of commands \NoAutoSpacing, \ttfamilyFB, \rmfamilyFB and \sffamilyFB have been completely redesigned in version 3.2c, they behave now consistently with all engines.

What's new in version 3.1?

New command \frquote{} meant to enter French quotations, especially long ones (spreading over several paragraphs) and/or embedded ones. see p. 3 for details.

What's new in version 3.0?

Many deep changes lead me to step babel-french's version number to 3.0a:

- Babel 3.9 is required now to process frenchb.ldf, this change allows for cleaner definitions of dates and captions for the Unicode engines LuaTeX and XeTeX and also provides a simpler syntax for end-users, see section 1.2.2 p.9.
- \frenchsetup{} options management has been completely reworked; two new options added.
- Canadian French didn't work as a normal Babel's dialect, it should now; btw. the French language should now be loaded as french, *not as* frenchb or francais and preferably as a *global* option of \documentclass. Some tolerance still exists in v3.0, but do not rely on it.
- babel-french no longer loads frenchb.cfg: customisation should definitely be done using \frenchsetup{} options.
- Description lists labels are now indented; try setting \descindentFB=0pt (or \listindentFB=0pt for all lists) in the preamble if you don't like it.
- The last but not least change affects the (recent) LuaTeX-based engines, (this means version 0.76 as included in TL2013 and up): active characters are no longer used in French for 'high punctuation' [11]. Functionalities and user interface are unchanged.

  Many thanks to Paul Isambert who provided the basis for the lua code (see his presentation at GUT'2010) and kindly reviewed my first drafts suggesting significant improvements.

Starting with version 3.0c, babel-french no longer customises lists with the beamer class and offers a new option (INGuillSpace) to follow French 'Imprimerie Nationale' recommendations regarding quotes' spacing.

[11] The current babel-french version requires LuaTeX v. 1.0.4 as included in TL2017, see above.

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.

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

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

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

Quit if eTeX is not available.

```
\let\bbl@tempa\relax
\begingroup\expandafter\expandafter\expandafter\endgroup
\expandafter\ifx\csname eTeXversion\endcsname\relax
  \let\bbl@tempa\endinput
  \fb@error{babel-french requires eTeX.\\
            Aborting here}
           {Orignal PlainTeX is not supported,\\
            please use LuaTeX or XeTeX engines.}
\fi
\bbl@tempa
```

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

```
\let\bbl@tempa\relax
\ifdefined\babeltags
\else
   \let\bbl@tempa\endinput
   \ifdefined\PackageError
      \PackageError{french.ldf}
         {babel-french requires babel v.3.16.\MessageBreak
          Aborting here}
         {Please upgrade Babel!}
   \else
```

```
      \fb@error{babel-french requires babel v.3.16.\\
                Aborting here}
               {Please upgrade Babel!}
  \fi
\fi
\bbl@tempa
```

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

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

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.

```
\ifdefined\l@acadian
  \adddialect\l@canadien\l@acadian
\else
  \adddialect\l@acadian\l@french
  \adddialect\l@canadien\l@french
\fi
```

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

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

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

\ifFBunicode
\ifFBLuaTeX
\ifFBXeTeX

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

```
\newif\ifFBunicode
\newif\ifFBLuaTeX
\newif\ifFBXeTeX
\begingroup\expandafter\expandafter\expandafter\endgroup
\expandafter\ifx\csname luatexversion\endcsname\relax
\else
  \FBunicodetrue \FBLuaTeXtrue
\fi
\begingroup\expandafter\expandafter\expandafter\endgroup
\expandafter\ifx\csname XeTeXrevision\endcsname\relax
\else
  \FBunicodetrue \FBXeTeXtrue
\fi
```

\ifFBfrench

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=»}.

```
\newif\ifFBfrench
```

\extrasfrench
\noextrasfrench

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

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

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.

```
\def\extrasfrench{%
    \FBfrenchtrue
    \babel@savevariable{\lccode"27}%
    \lccode"27="27
    \ifFBunicode
      \babel@savevariable{\lccode"2019}%
      \lccode"2019="2019
    \fi
}
\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.

```
\addto\extrasfrench{\bbl@frenchspacing}
\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.

```
\newif\ifFB@active@punct \FB@active@punctttrue
```

`\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).

```
\newif\ifFB@luatex@punct
\ifFBLuaTeX
  \ifnum\luatexversion<100
    \ifx\PackageWarning\@undefined
      \fb@warning{Please upgrade LuaTeX to version 1.0.4 or above!\\%
         babel-french will make high punctuation characters (;:!?)\\%
         active with LuaTeX < 1.0.4.}%
    \else
      \PackageWarning{french.ldf}{Please upgrade LuaTeX
         to version 1.0.4 or above!\MessageBreak
         babel-french will make high punctuation characters%
         \MessageBreak (;:!?) active with LuaTeX < 1.0.4;%
         \MessageBreak reported}%
    \fi
  \else
    \FB@luatex@punctttrue\FB@active@punctfalse
  \fi
\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.

```
\newcount\FB@stdchar
\newif\ifFB@xetex@punct
\ifdefined\XeTeXinterchartokenstate
  \FB@xetex@punctttrue\FB@active@punctfalse
  \ifdim\the\XeTeXversion\XeTeXrevision\p@ < 0.99994\p@
    \chardef\FB@nonchar="FF \relax
  \else
    \chardef\FB@nonchar="FFF \relax
  \fi
  \FB@stdchar=\z@
\fi
```

`\FBguillspace` `\FBcolonspace` `\FBthinspace` These three commands are meant for basic French. Other French dialects can use different settings, see below. According to the I.N. specifications, the ':' requires

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.

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

\FBsetspaces This command makes it easy to fine tune \FBguillspace, \FBcolonspace and \FBthinspace in French (defaut) 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.

```
\ifLaTeXe
  \newcommand*{\FBsetspaces}[5][french]{%
    \def\bbl@tempa{french}\def\bbl@tempb{#1}%
    \ifx\bbl@tempa\bbl@tempb \def\bbl@tempb{}\fi
    \@namedef{\bbl@tempb FB#2space}{\hskip #3\fontdimen2\font
                                      plus #4\fontdimen3\font
                                     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”.

```
    \ifFB@luatex@punct
      \ifx\bbl@tempb\FB@acadian
        \directlua{
          FBsp.#2.gl.ac[1] = #3
          FBsp.#2.gl.ac[2] = #4
          FBsp.#2.gl.ac[3] = #5
          if #3 > 0.6 then
             FBsp.#2.ch.ac = 0xA0
          elseif #3 > 0.2 then
             FBsp.#2.ch.ac = 0x202F
          else
             FBsp.#2.ch.ac = 0x200B
          end
        }%
      \fi
    \fi
```

```
  }
  \@onlypreamble\FBsetspaces
\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`.

```
\ifLaTeXe
  \addto\extrasfrench{%
    \ifFB@luatex@punct
      \edef\bbl@tempa{\detokenize\expandafter{\languagename}}%
      \edef\bbl@tempb{\detokenize{french}}%
      \ifx\bbl@tempa\bbl@tempb \FB@dialect=\z@
      \else                    \FB@dialect=\@ne
      \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!

```
      \ifdefined\FB@once\else
        \set@glue@table{colon}%
        \set@glue@table{thin}%
        \set@glue@table{guill}%
        \def\FB@once{}%
      \fi
    \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).

```
    \ifcsname\languagename FBthinspace\endcsname
      \babel@save\FBthinspace
      \renewcommand*{\FBthinspace}{%
              \csname\languagename FBthinspace\endcsname}%
    \fi
```

Same for `\FBcolonspace`:

```
    \ifcsname\languagename FBcolonspace\endcsname
      \babel@save\FBcolonspace
      \renewcommand*{\FBcolonspace}{%
              \csname\languagename FBcolonspace\endcsname}%
    \fi
```

And for `\FBguillspace`:

```
    \ifcsname\languagename FBguillspace\endcsname
      \babel@save\FBguillspace
      \renewcommand*{\FBguillspace}{%
```

```
                \csname\languagename FBguillspace\endcsname}%
    \fi
  }
\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.

```
\newif\ifFB@spacing \FB@spacingtrue
```

`\FB@spacing@off` `\FB@spacing@on` Two internal commands to switch on and off all space tuning for all six characters ‘;:!?«»’. 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).

```
\ifFB@luatex@punct
  \newcommand*{\FB@spacing@on}{\FB@spacing=\@ne}
  \newcommand*{\FB@spacing@off}{\FB@spacing=\z@}
\else
  \newcommand*{\FB@spacing@on}{\FB@spacingtrue}
  \newcommand*{\FB@spacing@off}{\FB@spacingfalse}
\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.

```
\ifFB@luatex@punct
  \ifdefined\newluafunction\else
```

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

```
    \input ltluatex.tex
  \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.

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

```
\ifLaTeXe
  \PackageInfo{french.ldf}{No need for active punctuation
               characters\MessageBreak with this version
               of LuaTeX!\MessageBreak reported}
\else
  \fb@info{No need for active punctuation characters\\
           with this version of LuaTeX!}
\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 `\FBsetspaces` commands) are taken into account. Values not explicitly set for Acadian by `\FBsetspaces[`*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.

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

`frenchb.lua` (*env.*) This is `frenchb.lua`. It holds Lua code to deal with ‘high punctuation’ and quotes. This code is based on suggestions from Paul Isambert.
First we define two flags to control spacing before French ‘high punctuation’ (thin space or inter-word space).

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

Managing spacing 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 ‘»’.

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

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

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

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

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

Local definitions for nodes:

```
local new_node     = node.new
local copy_node    = node.copy
local node_id      = node.id
local HLIST        = node_id("hlist")
local TEMP         = node_id("temp")
local KERN         = node_id("kern")
local GLUE         = node_id("glue")
local GLYPH        = node_id("glyph")
local PENALTY      = node_id("penalty")
```

```
local nobreak      = new_node(PENALTY)
nobreak.penalty    = 10000
local nbspace      = new_node(GLYPH)
local insert_node_before = node.insert_before
local insert_node_after  = node.insert_after
local remove_node        = node.remove
```

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

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

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

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

```
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.

```
local font_table = {}
local function new_glue_scaled (fid,table)
  if fid > 0 and table[1] then
     local fp = font_table[fid]
     if not fp then
        local ft = font.getfont(fid)
        if ft then
           font_table[fid] = ft.parameters
           fp = font_table[fid]
        end
     end
     local gl = new_node(GLUE,0)
     if fp then
        node.setglue(gl, table[1]*fp.space,
                         table[2]*fp.space_stretch,
                         table[3]*fp.space_shrink)
        return gl
     else
        return nil
     end
  else
     return nil
  end
end
```

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

```
local FBspacing    = luatexbase.attributes['FB@spacing']
local addDPspace   = luatexbase.attributes['FB@addDPspace']
local addGUILspace = luatexbase.attributes['FB@addGUILspace']
local FBucsNBSP    = luatexbase.attributes['FB@ucsNBSP']
local FBdialect    = luatexbase.attributes['FB@dialect']
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.

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

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

```
  node.kerning(head)
  for item in node.traverse_id(GLYPH, head) do
    local lang = item.lang
```

```
    local char = item.char
```

Skip glyphs not concerned by French kernings.

```
    if (lang == FR_fr or lang == FR_ca) and
         (FB_punct_left[char] or FB_punct_right[char]) then
       local fid  = item.font
       local attr = item.attr
       local FRspacing = has_attribute(item, FBspacing)
       FRspacing = FRspacing and FRspacing > 0
       local FRucsNBSP = has_attribute(item, FBucsNBSP)
       FRucsNBSP = FRucsNBSP and FRucsNBSP > 0
       local FRdialect = has_attribute(item, FBdialect)
       FRdialect = FRdialect and FRdialect > 0
       local SIG  = has_attribute(item, addGUILspace)
       SIG = SIG and SIG >0
       if FRspacing and fid > 0 then
          if FB_punct_left[char] then
             local prev = item.prev
             local prev_id, prev_subtype, prev_char
             if prev then
                prev_id = prev.id
                prev_subtype = prev.subtype
                if prev_id == GLYPH then
                   prev_char = prev.char
                end
             end
```

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

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

For characters for which `FB_punct_thin` or `FB_punct_thick` is *true*, the amount of spacing to be typeset before them is controlled by commands `\FBthinspace` and `\FBcolonspace` respectively. Two options: if a space has been typed in before (turned into *glue* in the node list), we remove the *glue* and add a nobreak penalty and the required *glue*. Otherwise (`auto` option), the penalty and the required *glue* are inserted if attribute `\FB@addDPspace` is set, unless any of these four conditions is met: a) node is ':' and the next one is of type GLYPH (avoids spurious spaces in http://mysite, C:\ or 10:35); b) the previous character is part of type `FB_punct_null` (avoids spurious spaces in strings like `(!)` or ??); c) a null glue (actually <= 1 sp for tabulars, possibly < 0) preceeds the punctuation character (for tabulars and listings); d) the punctuation character starts a paragraph or an `\hbox{}`.
When option `UnicodeNoBreakSpaces` is set to `true`, a Unicode character U+00A0 or U+202F is inserted instead of penalty and glue.

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

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

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

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

Let’s consider ‘»’ now (the only remaining glyph of `FB_punct_left` class): we just have

to remove any *glue* possibly preceeding ‘»’, 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 preceding glyph is member of FB_guil_null, or b) ‘»’ is the first glyph of an \hbox{} or a paragraph, nothing is done, this is controlled by the addgl flag.

```
            elseif SIG then
               local addgl = (prev_char and
                               not FB_guil_null[prev_char])
                              or
                              (not prev_char and
                               prev_id ~= TEMP and
                               not (prev_id == HLIST and
                                    prev_subtype == 3)
                              )
```

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

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

Similarly, for ‘«’ (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*.

```
         elseif SIG then
            local next = item.next
            local next_id, next_subtype, next_char, nextnext, kern_wd
            if next then
```

```
                next_id = next.id
                next_subtype = next.subtype
```

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

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

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

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

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

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

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

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

```
<*french>
\ifFB@luatex@punct
  \def\activate@luatexpunct{%
    \directlua{%
      FR_fr = \the\l@french ; FR_ca = \the\l@acadian ;
      local path = kpse.find_file("frenchb.lua", "lua")
```

```
      if path then
         local f = dofile(path)
         luatexbase.add_to_callback("kerning",
                     f, "frenchb.french_punctuation")
      else
         texio.write_nl('')
         texio.write_nl('*****************************')
         texio.write_nl('Error: frenchb.lua not found.')
         texio.write_nl('*****************************')
         texio.write_nl('')
      end
     }%
  }
\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 inital values will be restored when leaving French.

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

```
\ifFB@xetex@punct
  \ifLaTeXe
    \PackageInfo{french.ldf}{No need for active punctuation
                             characters\MessageBreak with this
                             version of XeTeX!\MessageBreak reported}
  \else
    \fb@info{No need for active punctuation characters\\
             with this version of XeTeX!}
  \fi
```

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

```
\newXeTeXintercharclass\FB@punctthick
\newXeTeXintercharclass\FB@punctthin
\newXeTeXintercharclass\FB@punctnul
\newXeTeXintercharclass\FB@guilo
\newXeTeXintercharclass\FB@guilf
\newXeTeXintercharclass\FB@guilnul
```

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

```
\def\FBsavevariable@loop#1#2{\begingroup
  \toks@\expandafter{\originalTeX #1}%
  \edef\x{\endgroup
    \def\noexpand\originalTeX{\the\toks@ #2=\the#1#2\relax}}%
  \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.

```
\def\FB@charlist{"21,"3A,"3B,"3F,"AB,"BB,"28,"5B,"A0,"202F}
\def\FB@charlistUCC{}
\def\FB@charlistxeCJK{}
\edef\FB@charlistsave{\FB@charlist}
\ifLaTeXe
  \AtBeginDocument{%
    \@ifpackageloaded{ucharclasses}%
      {\ifdefined\BasicLatinClass
         \RenewCommandCopy{\FB@stdchar}{\BasicLatinClass}%
         \def\FB@charlistUCC{"C0,"C2,"C6,"C7,"C8,"C9,"CA,"CB,"CE,"CF,%
           "D4,"D6,"D9,"DB,"DC,"E0,"E2,"E6,"E7,"E8,"E9,"EA,"EB,"EE,%
           "EF,"F4,"F6,"F9,"FB,"FC,"152,"153,"17F,"2019}%
         \addto\FB@charlist{,\FB@charlistUCC}%
         \edef\FB@charlistsave{\FB@charlist}%
       \fi
      }{}%
    \@ifpackageloaded{xeCJK}%
      {\def\FB@charlistxeCJK{%
               "29,"5D,"7B,"7D,"2C,"2D,"2E,"22,"25,"27,"60,"2019}%
       \addto\FB@charlist{,\FB@charlistxeCJK}%
       \edef\FB@charlistsave{\FB@charlist}%
      }{}%
  }
\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.

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

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

```
\ifx\FB@charlistUCC\@empty\else
   \bbl@for\FB@char\FB@charlistUCC
     {\XeTeXcharclass \FB@char \FB@stdchar}%
\fi
```

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

```
\ifx\FB@charlistxeCJK\@empty\else
  \bbl@for\FB@char\FB@charlistxeCJK
          {\XeTeXcharclass\FB@char=\FB@stdchar}%
\fi
```

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

```
\bbl@for\FB@char {\`\[,\`\(,"A0,"202F}%
        {\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).

```
\XeTeXinterchartokenstate=\@ne
\XeTeXcharclass `\: = \FB@punctthick
\XeTeXinterchartoks \FB@stdchar \FB@punctthick = {%
      \ifFB@spacing\ifhmode\FDP@colonspace\fi\fi}%
\XeTeXinterchartoks \FB@guilf \FB@punctthick = {%
      \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.

```
\XeTeXinterchartoks \FB@nonchar \FB@punctthick = {%
      \ifFB@spacing
        \ifhmode
          \ifdim\lastskip>1sp
            \unskip\penalty\@M\FBcolonspace
          \else
            \FDP@colonspace
          \fi
        \fi
      \fi}%
\bbl@for\FB@char {\`\;,\`\!,\`\?}%
        {\XeTeXcharclass\FB@char=\FB@punctthin}%
\XeTeXinterchartoks \FB@stdchar \FB@punctthin = {%
      \ifFB@spacing\ifhmode\FDP@thinspace\fi\fi}%
\XeTeXinterchartoks \FB@guilf \FB@punctthin = {%
      \ifFB@spacing\FDP@thinspace\fi}%
\XeTeXinterchartoks \FB@nonchar \FB@punctthin = {%
      \ifFB@spacing
        \ifhmode
          \ifdim\lastskip>1sp
            \unskip\penalty\@M\FBthinspace
```

```
              \else
                \FDP@thinspace
              \fi
            \fi
          \fi}%
    \XeTeXinterchartoks \FB@guilo \FB@stdchar = {%
          \ifFB@spacing\FB@guillspace\fi}%
    \XeTeXinterchartoks \FB@guilo \FB@nonchar = {%
          \ifFB@spacing\FB@guillspace\ignorespaces\fi}%
    \XeTeXinterchartoks \FB@stdchar \FB@guilf = {%
          \ifFB@spacing\FB@guillspace\fi}%
    \XeTeXinterchartoks \FB@punctthin \FB@guilf = {%
          \ifFB@spacing\FB@guillspace\fi}%
    \XeTeXinterchartoks \FB@nonchar \FB@guilf = {%
          \ifFB@spacing\unskip\FB@guillspace\fi}%
  }
  \addto\extrasfrench{\FB@xetex@punct@french}
```

End of specific code for punctuation with modern XeTeX engines.

```
\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 `:`.

```
\newif\ifFB@koma
\ifLaTeXe
  \@ifclassloaded{scrartcl}{\FB@komatrue}{}
  \@ifclassloaded{scrbook}{\FB@komatrue}{}
  \@ifclassloaded{scrreprt}{\FB@komatrue}{}
  \ifFB@koma\def\FB@std@capsep{:\ }\fi
  \@ifclassloaded{beamer}{\def\FB@std@capsep{:\ }}{}
  \@ifclassloaded{memoir}{\def\FB@std@capsep{: }}{}
\fi
```

```
\ifFB@active@punct
  \initiate@active@char{:}%
  \initiate@active@char{;}%
  \initiate@active@char{!}%
  \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`.

```
  \declare@shorthand{french}{;}{%
    \ifFB@spacing
      \ifhmode
        \ifdim\lastskip>1sp
          \unskip\penalty\@M\FBthinspace
```

```
        \else
          \FDP@thinspace
        \fi
      \fi
    \fi
```

Now we can insert a ; character.

```
    \string;}
```

The next three definitions are very similar.

```
  \declare@shorthand{french}{!}{%
    \ifFB@spacing
      \ifhmode
        \ifdim\lastskip>1sp
          \unskip\penalty\@M\FBthinspace
        \else
          \FDP@thinspace
        \fi
      \fi
    \fi
    \string!}
  \declare@shorthand{french}{?}{%
    \ifFB@spacing
      \ifhmode
        \ifdim\lastskip>1sp
          \unskip\penalty\@M\FBthinspace
        \else
          \FDP@thinspace
        \fi
      \fi
    \fi
    \string?}
  \declare@shorthand{french}{:}{%
    \ifFB@spacing
      \ifhmode
        \ifdim\lastskip>1sp
          \unskip\penalty\@M\FBcolonspace
        \else
          \FDP@colonspace
        \fi
      \fi
    \fi
    \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.

```
  \declare@shorthand{system}{:}{\string:}
  \declare@shorthand{system}{!}{\string!}
  \declare@shorthand{system}{?}{\string?}
  \declare@shorthand{system}{;}{\string;}
```

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

```
  \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.

```
    \bbl@activate{:}\bbl@activate{;}%
    \bbl@activate{!}\bbl@activate{?}%
  }
  \addto\noextrasfrench{%
    \bbl@deactivate{:}\bbl@deactivate{;}%
    \bbl@deactivate{!}\bbl@deactivate{?}%
  }
\fi
```

2.2.4 Punctuation switches common to all engines

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

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

`\AutoSpaceBeforeFDP` `\NoAutoSpaceBeforeFDP`

`\autospace@beforeFDP` and `\noautospace@beforeFDP` are internal commands. `\autospace@beforeFDP` defines `\FDP@thinspace` and `\FDP@colonspace` as non-breaking spaces and sets LuaTeX attribute `\FB@addDPspace` to 1 (true), while `\noautospace@beforeFDP` lets these spaces empty and sets flag `\FB@addDPspace` to 0 (false). User commands `\AutoSpaceBeforeFDP` and `\NoAutoSpaceBeforeFDP` do the same and take care of the flag `\ifFBAutoSpacePunctuation` in LaTeX.
Set the default now for Plain (done later for LaTeX).

```
\def\autospace@beforeFDP{%
  \ifFB@luatex@punct \FB@addDPspace=\@ne \fi
  \def\FDP@thinspace{\penalty\@M\FBthinspace}%
  \def\FDP@colonspace{\penalty\@M\FBcolonspace}}
\def\noautospace@beforeFDP{%
  \ifFB@luatex@punct \FB@addDPspace=\z@ \fi
  \let\FDP@thinspace\@empty
  \let\FDP@colonspace\@empty}
\ifLaTeXe
  \def\AutoSpaceBeforeFDP{\autospace@beforeFDP
                          \FBAutoSpacePunctuationtrue}
  \def\NoAutoSpaceBeforeFDP{\noautospace@beforeFDP
                            \FBAutoSpacePunctuationfalse}
  \AtEndOfPackage{\AutoSpaceBeforeFDP}
\else
  \let\AutoSpaceBeforeFDP\autospace@beforeFDP
  \let\NoAutoSpaceBeforeFDP\noautospace@beforeFDP
  \AutoSpaceBeforeFDP
\fi
```

`\rmfamilyFB` `\sffamilyFB` `\ttfamilyFB`

In LaTeX2e `\ttfamily` (and hence `\texttt`) will be redefined ‘AtBeginDocument’ as `\ttfamilyFB` so that no space is added before the four `;` `:` `!` `?` characters, even if `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.

```
\ifLaTeXe
  \DeclareRobustCommand\ttfamilyFB{\FB@spacing@off \ttfamilyORI}
  \DeclareRobustCommand\rmfamilyFB{\FB@spacing@on  \rmfamilyORI}
  \DeclareRobustCommand\sffamilyFB{\FB@spacing@on  \sffamilyORI}
\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.

```
\DeclareRobustCommand*{\NoAutoSpacing}{%
  \FB@spacing@off
  \ifFB@active@punct\shorthandoff{;:!?}\fi
}
```

2.3 Commands for French quotation marks

`\guillemotleft` `\guillemotright` `\textquoteddblleft` `\textquoteddblright` pdfLaTeX users are supposed to use 8-bit output encodings (T1, LY1,...) to typeset French, those who still stick to OT1 should load `aeguill` or a similar package. In both cases the commands `\guillemotleft` and `\guillemotright` will print the 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.

```
\ifLaTeXe
\else
  \ifFBunicode
    \def\guillemotleft{{\char"00AB}}
    \def\guillemotright{{\char"00BB}}
    \def\textquotedblleft{{\char"201C}}
    \def\textquotedblright{{\char"201D}}
  \else
    \def\guillemotleft{\leavevmode\raise0.25ex
                       \hbox{$\scriptscriptstyle\ll$}}
    \def\guillemotright{\raise0.25ex
                        \hbox{$\scriptscriptstyle\gg$}}
    \def\textquotedblleft{``}
    \def\textquotedblright{''}
  \fi
  \let\xspace\relax
\fi
```

\FBgspchar
\FB@og
\FB@fg

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 (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.

```
\providecommand\texorpdfstring[2]{#1}
\newcommand*{\FB@og}{\texorpdfstring{\@FB@og}{\guillemotleft\space}}
\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.

```
\newcommand*{\FB@guillspace}{\penalty\@M\FBguillspace}
\newcommand*{\FBgspchar}{\char"A0\relax}
\newif\ifFBucsNBSP
\ifFB@luatex@punct
  \DeclareRobustCommand*{\@FB@og}{\leavevmode
          \bgroup\FB@spacing=\z@ \guillemotleft\egroup
          \ifFBucsNBSP\FBgspchar\else\FB@guillspace\fi}
  \DeclareRobustCommand*{\@FB@fg}{\ifdim\lastskip>\z@\unskip\fi
          \ifFBucsNBSP\FBgspchar\else\FB@guillspace\fi
          \bgroup\FB@spacing=\z@ \guillemotright\egroup}
\fi
```

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

```
\ifFB@xetex@punct
  \DeclareRobustCommand*{\@FB@og}{\leavevmode
        \bgroup\FB@spacingfalse\guillemotleft\egroup
        \FB@guillspace}
  \DeclareRobustCommand*{\@FB@fg}{\ifdim\lastskip>\z@\unskip\fi
        \FB@guillspace
        \bgroup\FB@spacingfalse\guillemotright\egroup}
\fi
\ifFB@active@punct
  \DeclareRobustCommand*{\@FB@og}{\leavevmode
        \guillemotleft
        \FB@guillspace}
  \DeclareRobustCommand*{\@FB@fg}{\ifdim\lastskip>\z@\unskip\fi
        \FB@guillspace
        \guillemotright}
\fi
```

\og
\fg

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.

```
\newcommand*{\og}{\@empty}
\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).

```
\ifLaTeXe
  \def\bbl@frenchguillemets{%
        \renewcommand*{\og}{\FB@og}%
        \renewcommand*{\fg}{\FB@fg\xspace}}
  \renewcommand*{\og}{\textquotedblleft}
  \renewcommand*{\fg}{\ifdim\lastskip>\z@\unskip\fi
                      \textquotedblright\xspace}
\else
  \def\bbl@frenchguillemets{\let\og\FB@og
                            \let\fg\FB@fg}
  \def\og{\textquotedblleft}
  \def\fg{\ifdim\lastskip>\z@\unskip\fi\textquotedblright}
\fi

\addto\extrasfrench{\babel@save\og \babel@save\fg
                    \bbl@frenchguillemets}
```

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

```
\newcommand*{\ogi}{\FB@og}
\newcommand*{\fgi}{\FB@fg}
\newcommand*{\@ogi}{\ifmmode\hbox{\ogi}\else\ogi\fi}
\newcommand*{\@fgi}{\ifmmode\hbox{\fgi}\else\fgi\fi}
\newcommand*{\ogii}{\textquotedblleft}
\newcommand*{\fgii}{\textquotedblright}
\newcommand*{\@ogii}{\ifmmode\hbox{\ogii}\else\ogii\fi}
\newcommand*{\@fgii}{\ifmmode\hbox{\fgii}\else\fgii\fi}
```

and the needed technical stuff to handle options:

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

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

```
\def\FB@addquote@everypar{%
  \let\FBnew@everypar\everypar
  \FBold@everypar=\expandafter{\the\everypar}%
  \FBnew@everypar={\the\FBold@everypar\FBeverypar@quote}%
  \let\everypar\FBold@everypar
  \let\FB@addquote@everypar\relax
}
\newif\ifFBcloseguill  \FBcloseguilltrue
\newif\ifFBInnerGuillSingle
\def\FBguillopen{\bgroup\NoAutoSpacing\guillemotleft\egroup}
\def\FBguillclose{\bgroup\NoAutoSpacing\guillemotright\egroup}
\let\FBguillnone\empty
\let\FBeveryparguill\FBguillopen
\let\FBeverylineguill\FBguillnone
\let\FBeverypar@quote\relax
\let\FBeveryline@quote\empty
```

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

```
\ifLaTeXe
  \DeclareRobustCommand\frquote{%
    \texorpdfstring{\@ifstar{\FBcloseguillfalse\fr@quote}%
                            {\FBcloseguilltrue \fr@quote}}%
                   {\bm@fr@quote}%
  }
  \newcommand{\bm@fr@quote}[1]{%
    \guillemotleft\space #1\space\guillemotright}
\else
  \newcommand\frquote[1]{\fr@quote{#1}}
\fi
```

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

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

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

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

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

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

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

```
        \let\FBeverypar@quote\relax
        \ifFBInnerGuillSingle
```

```
          \def\ogii{\leavevmode
                    \guilsinglleft\FB@guillspace}%
          \def\fgii{\ifdim\lastskip>\z@\unskip\fi
                    \FB@guillspace\guilsinglright}%
          \ifx\FBeveryparguill\FBguillopen
            \def\FBeverypar@quote{\guilsinglleft\FB@guillspace}%
          \fi
          \ifx\FBeveryparguill\FBguillclose
            \def\FBeverypar@quote{\guilsinglright\FB@guillspace}%
          \fi
        \fi
        \@ogii #1\ifFBcloseguill \@fgii \fi
      \fi
    \fi
  \else
```

Warn if `\FBguill@level` > 2:

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

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

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

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

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

2.4 Date in French

`\frenchtoday` `\frenchdate` `\datefrench`

The following code creates a macro `\datefrench` which in turn defines command `\frenchtoday` (`\today` is defined as `\frenchtoday` in French). The corresponding 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.ldf` 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.

```
\def\BabelLanguages{french,acadian}
\StartBabelCommands*{\BabelLanguages}{date}
   [unicode, fontenc=TU EU1 EU2, charset=utf8]
  \SetString\monthiiname{février}
  \SetString\monthviiiname{août}
  \SetString\monthxiiname{décembre}
\StartBabelCommands*{\BabelLanguages}{date}
  \SetStringLoop{month#1name}{%
      janvier,f\'evrier,mars,avril,mai,juin,juillet,%
      ao\^ut,septembre,octobre,novembre,d\'ecembre}
  \SetString\today{\FB@date{\year}{\month}{\day}}
\EndBabelCommands
```

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

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

2.5 Extra utilities

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

`\up` `\fup`

`\up` eases the typesetting of superscripts like '1er'. 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 `scalefnt` which will be loaded at the end of Babel's loading (`babel-french` being an option of Babel, it cannot load a package while being read).

```
\newif\ifFB@poorman
\newdimen\FB@Mht
\ifLaTeXe
  \AtEndOfPackage{\RequirePackage{scalefnt}}
```

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

```
  \newcommand*{\FBsupR}{-0.12}
```

```
\newcommand*{\FBsupS}{0.65}
\newcommand*{\FB@lc}[1]{\MakeLowercase{#1}}
\DeclareRobustCommand*{\FB@up@fake}[1]{%
  \settoheight{\FB@Mht}{M}%
  \addtolength{\FB@Mht}{\FBsupR \FB@Mht}%
  \addtolength{\FB@Mht}{-\FBsupS ex}%
  \raisebox{\FB@Mht}{\scalefont{\FBsupS}{\FB@lc{#1}}}%
  }
```

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.

```
\def\FB@split#1#2#3#4\@nil{\def\FB@firstthree{#1#2#3}%
                           \def\FB@suffix{#4}}
\def\FB@x{x}
\def\FB@j{j}
\DeclareRobustCommand*{\FB@up}[1]{%
  \bgroup \FB@poormantrue
    \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.

```
    \edef\reserved@a{\lowercase{%
       \noexpand\IfFileExists{\f@encoding\FB@firstthree -sup.fd}}}%
    \reserved@a
      {\ifx\FB@suffix\FB@x \FB@poormanfalse\fi
       \ifx\FB@suffix\FB@j \FB@poormanfalse\fi
       \ifFB@poorman \FB@up@fake{#1}%
       \else         \FB@up@real{#1}%
       \fi}%
      {\FB@up@fake{#1}}%
  \egroup}
```

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

```
\newcommand*{\FB@up@real}[1]{\bgroup
     \fontfamily{\FB@firstthree -sup}\selectfont \FB@lc{#1}\egroup}
```

`\fup` is defined as `\FB@up` unless `\realsuperscript` is defined by `realscripts.sty`. `\fup` just prints its argument in bookmarks.

```
\DeclareRobustCommand*{\fup}[1]{%
  \texorpdfstring{\ifx\realsuperscript\@undefined
                    \FB@up{#1}%
                  \else
```

```
                        \bgroup\let\fakesuperscript\FB@up@fake
                          \realsuperscript{\FB@lc{#1}}\egroup
                      \fi
                      }{#1}%
  }
```

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

```
  \providecommand*{\up}{\fup}
```

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

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

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

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

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

`\FBmedkern` `\FBthickkern`

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

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

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

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

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

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

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

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

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

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

```
\DeclareRobustCommand*{\Nos}{%
    \texorpdfstring{N\up{os}\FBmedkern}{N\textdegree\space}}
\DeclareRobustCommand*{\nos}{%
    \texorpdfstring{n\up{os}\FBmedkern}{n\textdegree\space}}
```

\bname
\bsc

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

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

Some definitions for special characters. We won't define `\tilde` as a Text Symbol not to conflict with the macro `\tilde` for math mode and use the name `\tild` instead. Note that `\boi` may *not* be used in math mode, its name in math mode is `\backslash`. `\degre` can be accessed by the command `\r{}` for ring accent.

```
\ifFBunicode
  \providecommand*{\textbackslash}{{\char"005C}}
  \providecommand*{\textasciicircum}{{\char"005E}}
  \providecommand*{\textasciitilde}{{\char"007E}}
  \newcommand*{\FB@degre}{°}
\else
  \ifLaTeXe
    \newcommand*{\FB@degre}{\r{}}
  \fi
\fi
\DeclareRobustCommand*{\boi}{\textbackslash}
\DeclareRobustCommand*{\circonflexe}{\textasciicircum}
\DeclareRobustCommand*{\tild}{\textasciitilde}
\DeclareRobustCommand*{\degre}{%
  \texorpdfstring{\FB@degre}{\textdegree}}
\newcommand*{\at}{@}
```

\degres

We now define a macro `\degres` for typesetting the abbreviation for 'degrees' (as in 'degrees Celsius'). As the bounding box of the character 'degree' has *very* different widths in CM/EC and PostScript fonts, we fix the width of the bounding box of `\degres` to 0.3 em, this lets the symbol 'degree' stick to the preceding (e.g., 45`\degres`) or following character (e.g., 20~`\degres` C). `\degres` works in math-mode (angles).
If TEX Companion fonts are available (`textcomp.sty`), we pick up `\textdegree` from them instead of emulating 'degrees' from the `\r{}` accent. Otherwise we advise the user (once only) to use TS1-encoding.

```
\DeclareRobustCommand*{\degres}{\degre}
\ifLaTeXe
  \AtBeginDocument{%
    \@ifpackageloaded{fontspec}{}{%
      \ifdefined\DeclareEncodingSubset
        \DeclareRobustCommand*{\degres}{%
          \texorpdfstring{\hbox{\UseTextSymbol{TS1}{\textdegree}}}%
                         {\textdegree}}%
      \else
        \def\Warning@degree@TSone{\FBWarning
             {Degrees would look better in TS1-encoding:%
              \MessageBreak add \protect
              \usepackage{textcomp} to the preamble.%
              \MessageBreak Degrees used}}
        \DeclareRobustCommand*{\degres}{%
          \texorpdfstring{\hbox to 0.3em{\hss\degre\hss}%
                         \Warning@degree@TSone
                         \global\let\Warning@degree@TSone\relax}%
                         {\textdegree}}%
      \fi
    }%
  }
\fi
```

2.6 Formatting numbers

`\StandardMathComma` `\DecimalMathComma`

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

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

```
  \std@mcc=\mathcode`\,
  \dec@mcc=\std@mcc
  \@tempcnta=\std@mcc
  \divide\@tempcnta by "1000
  \multiply\@tempcnta by "1000
  \advance\dec@mcc by -\@tempcnta
  \newcommand*{\dec@math@comma}{\mathcode`\,=\dec@mcc}
  \newcommand*{\std@math@comma}{\mathcode`\,=\std@mcc}
\fi
\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`.

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

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

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

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

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

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

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

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

```
\AtBeginDocument{%
  \@FBpreamblefalse
  \@ifpackageloaded{icomma}%
     {\FB@icommatrue
      \if@preamble@DecimalMathComma
        \PackageWarning{french.ldf}{%
          icomma package loaded, \protect\DecimalMathComma%
          \MessageBreak does nothing.  Reported}%
      \fi
     }%
     {\if@preamble@DecimalMathComma
        \ifFB@mainlanguage@FR \dec@math@comma \fi
        \let\dec@m@c\dec@math@comma
        \addto\extrasfrench{\dec@m@c}%
        \ifdefined\extrasacadian
          \addto\extrasacadian{\dec@m@c}%
        \fi
      \fi
```

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

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

```
      \fi
    }%
  }
```

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

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

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

2.7 Caption names

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

\captionsfrench Let's first define \captionsfrench 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.

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

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

```
\StartBabelCommands*{\BabelLanguages}{captions}
      [unicode, fontenc=TU EU1 EU2, charset=utf8]
  \SetString{\refname}{Références}
  \SetString{\abstractname}{Résumé}
  \SetString{\prefacename}{Préface}
  \SetString{\contentsname}{Table des matières}
  \SetString{\ccname}{Copie à }
  \SetString{\proofname}{Démonstration}
  \SetString{\partfirst}{Première}
  \SetString{\partsecond}{Deuxième}
  \SetStringLoop{ordinal#1}{%
    \frenchpartfirst,\frenchpartsecond,Troisième,Quatrième,%
```

```
    Cinquième,Sixième,Septième,Huitième,Neuvième,Dixième,Onzième,%
    Douzième,Treizième,Quatorzième,Quinzième,Seizième,%
    Dix-septième,Dix-huitième,Dix-neuvième,Vingtième}
\StartBabelCommands*{\BabelLanguages}{captions}
  \SetString{\refname}{R\'ef\'erences}
  \SetString{\abstractname}{R\'esum\'e}
  \SetString{\bibname}{Bibliographie}
  \SetString{\prefacename}{Pr\'eface}
  \SetString{\chaptername}{Chapitre}
  \SetString{\appendixname}{Annexe}
  \SetString{\contentsname}{Table des mati\`eres}
  \SetString{\listfigurename}{Table des figures}
  \SetString{\listtablename}{Liste des tableaux}
  \SetString{\indexname}{Index}
  \SetString{\figurename}{Figure}
  \SetString{\tablename}{Table}
  \SetString{\pagename}{page}
  \SetString{\seename}{voir}
  \SetString{\alsoname}{voir aussi}
  \SetString{\enclname}{P.~J. }
  \SetString{\ccname}{Copie \`a }
  \SetString{\headtoname}{}
  \SetString{\proofname}{D\'emonstration}
  \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`.

```
  \SetString{\partfirst}{Premi\`ere}
  \SetString{\partsecond}{Deuxi\`eme}
  \SetString{\partnameord}{partie}
  \SetStringLoop{ordinal#1}{%
    \partfirst,\partsecond,Troisi\`eme,Quatri\`eme, Cinqui\`eme,%
    Sixi\`eme,Septi\`eme,Huiti\`eme,Neuvi\`eme,Dixi\`eme,%
    Onzi\`eme,Douzi\`eme,Treizi\`eme,Quatorzi\`eme,Quinzi\`eme,%
    Seizi\`eme,Dix-septi\`eme,Dix-huiti\`eme,Dix-neuvi\`eme,%
    Vingti\`eme}
  \AfterBabelCommands{%
    \DeclareRobustCommand*{\FB@emptypart}{\def\thepart{\unskip}}%
    \DeclareRobustCommand*{\FB@partname}{%
       \ifFBPartNameFull
         \csname ordinal\romannumeral\value{part}\endcsname\space
         \partnameord\FB@emptypart
       \else
         Partie%
       \fi}%
   }
  \SetString{\partname}{\FB@partname}
\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.).

```
\AtBeginDocument{%
```

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

2.8 Figure and table captions

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

```
\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 preceed 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 \@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.ldf makes '>' active).

```
\bgroup
  \catcode`:=12 \catcode`>=12 \relax
  \long\gdef\STD@makecaption#1#2{%
    \vskip\abovecaptionskip
    \sbox\@tempboxa{#1: #2}%
    \ifdim \wd\@tempboxa >\hsize
      #1: #2\par
    \else
      \global \@minipagefalse
      \hb@xt@\hsize{\hfil\box\@tempboxa\hfil}%
    \fi
    \vskip\belowcaptionskip}
\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.

```
\newif\if@FBwarning@capsep
\ifFB@active@punct\@FBwarning@capseptrue\fi
\newcommand*{\CaptionSeparator}{\space\textendash\space}
\def\FBCaption@Separator{: }
\long\def\FB@makecaption#1#2{%
  \vskip\abovecaptionskip
  \sbox\@tempboxa{#1\FBCaption@Separator #2}%
  \ifdim \wd\@tempboxa >\hsize
    #1\FBCaption@Separator #2\par
  \else
    \global \@minipagefalse
    \hb@xt@\hsize{\hfil\box\@tempboxa\hfil}%
  \fi
  \vskip\belowcaptionskip}
```

Disable the standard warning with AMS and SMF classes.

```
\@ifclassloaded{amsart}{\@FBwarning@capsepfalse}{}
\@ifclassloaded{amsbook}{\@FBwarning@capsepfalse}{}
\@ifclassloaded{amsdtx}{\@FBwarning@capsepfalse}{}
\@ifclassloaded{amsldoc}{\@FBwarning@capsepfalse}{}
\@ifclassloaded{amproc}{\@FBwarning@capsepfalse}{}
\@ifclassloaded{smfart}{\@FBwarning@capsepfalse}{}
\@ifclassloaded{smfbook}{\@FBwarning@capsepfalse}{}
```

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

```
\@ifclassloaded{IEEEconf}{\@FBwarning@capsepfalse}{}
\@ifclassloaded{IEEEtran}{\@FBwarning@capsepfalse}{}
\@ifclassloaded{revtex4-2}{\@FBwarning@capsepfalse}{}
\@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`)

```
\@ifclassloaded{memoir}{\@FBwarning@capsepfalse}{}
\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`).

```
\@ifclassloaded{beamer}{\@FBwarning@capsepfalse}{}
\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 `\FBCaption@Separator`, taking care that the colon is typeset correctly in French (*not* 'Figure 1: légende').

```
\AtBeginDocument{%
  \ifx\@makecaption\STD@makecaption
    \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`.

```
    \ifFBOldFigTabCaptions
    \else
      \def\FBCaption@Separator{{\autospace@beforeFDP : }}%
      \ifFBCustomiseFigTabCaptions
        \ifFB@mainlanguage@FR
          \def\FBCaption@Separator{\CaptionSeparator}%
        \fi
      \fi
    \fi
    \@FBwarning@capsepfalse
  \fi
```

No Warning if `caption.sty` or `caption-light.sty` has been loaded.

```
    \@ifpackageloaded{caption}{\@FBwarning@capsepfalse}{}%
    \@ifpackageloaded{caption-light}{\@FBwarning@capsepfalse}{}%
```

Final warning if relevant:

```
  \if@FBwarning@capsep
     \FBWarning
       {Figures' and tables' captions might look like\MessageBreak
        `Figure 1:' in French instead of `Figure 1 :'.\MessageBreak
        If this happens, to fix this issue\MessageBreak
        switch to LuaLaTeX or XeLaTeX or\MessageBreak
        try to add \protect\usepackage{caption} or\MessageBreak
        ... leave it as it is; reported}%
  \fi
  \let\FB@makecaption\relax
  \let\STD@makecaption\relax
}
```

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.

```
\ifFBunicode
\else
  \DeclareTextSymbol{\FBtextellipsis}{LY1}{133}
  \DeclareTextCommand{\FBtextellipsis}{PU}{\9040\046}
  \DeclareTextCommand{\FBtextellipsis}{PD1}{\203}
  \DeclareTextCommandDefault{\FBtextellipsis}{%
```

```
      .\kern\fontdimen3\font.\kern\fontdimen3\font.\xspace}%
  \def\bbl@frenchdots{\babel@save\textellipsis
                      \let\textellipsis\FBtextellipsis}
  \addto\extrasfrench{\bbl@frenchdots}
\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).

```
\ifFB@active@punct
   \@ifpackageloaded{listings}
      {\AtBeginDocument{%
         \FBWarning{Please load the "listings" package\MessageBreak
                    AFTER babel/french; reported}}%
      }{}
\fi
```

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

```
\newif\if@FBwarning@natbib
\ifFB@active@punct
  \@ifpackageloaded{natbib}{}{\@FBwarning@natbibtrue}
\fi
\AtBeginDocument{%
   \if@FBwarning@natbib
     \@ifpackageloaded{natbib}{}{\@FBwarning@natbibfalse}%
   \fi
   \if@FBwarning@natbib
     \FBWarning{Please load the "natbib" package\MessageBreak
                BEFORE babel/french; reported}%
   \fi
}
```

Package `beamerarticle` should be loaded before `babel-french` to avoid list's conflicts, see p. 55.

```
\newif\if@FBwarning@beamerarticle
\@ifpackageloaded{beamerarticle}{}{\@FBwarning@beamerarticletrue}
\AtBeginDocument{%
   \if@FBwarning@beamerarticle
     \@ifpackageloaded{beamerarticle}{}%
                                      {\@FBwarning@beamerarticlefalse}%
   \fi
   \if@FBwarning@beamerarticle
     \FBWarning{Please load the "beamerarticle" package\MessageBreak
                BEFORE babel/french; reported}%
   \fi
}
```

2.11 Setup options: keyval stuff

All setup options are handled by command `\frenchsetup{}` using the keyval syntax. A list of flags is defined and set to a default value which will possibly be changed 'AtEnd-

OfPackage' if French is the main language. After this, \frenchsetup{} eventually modifies the preset values of these flags.
Option processing can occur either in \frenchsetup{}, but *only for options explicitly set* by \frenchsetup{}, or 'AtBeginDocument'; any option affecting \extrasfrench{} *must* be 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 an option which was used up to v2.6h, it has been dropped in v3.0a because of its side-effects (f.i. \babel@save and \babel@savevariable did not work for French).

\frenchsetup 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.

```
\newcommand*{\frenchsetup}[1]{%
  \setkeys{FB}{#1}%
}%
\@onlypreamble\frenchsetup
```

Keep the former name \frenchbsetup working for compatibility.

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

We define a collection of conditionals with their defaults (true or false).

```
\newif\ifFBShowOptions
\newif\ifFBStandardLayout           \FBStandardLayouttrue
\newif\ifFBGlobalLayoutFrench       \FBGlobalLayoutFrenchtrue
\newif\ifFBReduceListSpacing
\newif\ifFBStandardListSpacing      \FBStandardListSpacingtrue
\newif\ifFBListOldLayout
\newif\ifFBListItemsAsPar
\newif\ifFBCompactItemize
\newif\ifFBStandardItemizeEnv       \FBStandardItemizeEnvtrue
\newif\ifFBStandardEnumerateEnv     \FBStandardEnumerateEnvtrue
\newif\ifFBStandardItemLabels       \FBStandardItemLabelstrue
\newif\ifFBStandardLists            \FBStandardListstrue
\newif\ifFBIndentFirst
\newif\ifFBFrenchFootnotes
\newif\ifFBAutoSpaceFootnotes
\newif\ifFBOriginalTypewriter
\newif\ifFBThinColonSpace
\newif\ifFBThinSpaceInFrenchNumbers
\newif\ifFBFrenchSuperscripts       \FBFrenchSuperscriptstrue
\newif\ifFBLowercaseSuperscripts    \FBLowercaseSuperscriptstrue
\newif\ifFBPartNameFull             \FBPartNameFulltrue
\newif\ifFBCustomiseFigTabCaptions
\newif\ifFBOldFigTabCaptions
\newif\ifFBSmallCapsFigTabCaptions  \FBSmallCapsFigTabCaptionstrue
\newif\ifFBSuppressWarning
\newif\ifFBINGuillSpace
```

The defaults values of these flags have been choosen so that babel-french does not change anything regarding the global layout. \bbl@main@language, set by the last

option of Babel, controls the global layout of the document. ‘AtEndOfPackage’ we check the main language in \bbl@main@language; if it is French (or a French dialect) the values of some flags have to be changed to ensure a French looking layout for the whole document (even in parts written in languages other than French); the end-user will then be able to customise the values of all these flags with \frenchsetup{}.
The following patch is for koma-script classes: the \partformat command, defined as \partname~\thepart\autodot, is incompatible with our redefinition of \partname.

```
\ifFB@koma
  \ifdefined\partformat
    \def\FB@partformat@fix{%
           \ifFBPartNameFull
             \babel@save\partformat
             \renewcommand*{\partformat}{\partname}%
           \fi}
    \addto\extrasfrench{\FB@partformat@fix}%
  \fi
\fi
```

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.

```
\def\FB@french{french}
\def\FB@acadian{acadian}
\newif\ifFB@mainlanguage@FR
\AtEndOfPackage{%
  \ifx\bbl@main@language\FB@french \FB@mainlanguage@FRtrue
  \else \ifx\bbl@main@language\FB@acadian \FB@mainlanguage@FRtrue \fi
  \fi
  \ifFB@mainlanguage@FR
    \FBGlobalLayoutFrenchtrue
    \@ifclassloaded{beamer}%
      {\PackageInfo{french.ldf}{%
          No list customisation for the beamer class,%
          \MessageBreak reported}}%
      {\@ifpackageloaded{beamerarticle}%
         {\FBStandardItemLabelsfalse
          \FBStandardListSpacingfalse
          \PackageInfo{french.ldf}{%
             Minimal list customisation for the beamerarticle%
             \MessageBreak package; reported}}%
```

Otherwise customise lists “à la française’’:

```
         {\FBStandardListSpacingfalse
          \FBStandardItemizeEnvfalse
          \FBStandardEnumerateEnvfalse
          \FBStandardItemLabelsfalse}%
      }
    \FBIndentFirsttrue
    \FBFrenchFootnotestrue
    \FBAutoSpaceFootnotestrue
```

```
    \FBCustomiseFigTabCaptionstrue
  \fi
```

babel-french being an option of Babel, it cannot load a package (keyval) while french.ldf is read, so we defer the loading of keyval and the options setup at the end of Babel's loading.

```
  \RequirePackage{keyval}%
  \define@key{FB}{ShowOptions}[true]%
          {\csname FBShowOptions#1\endcsname}%
```

The next two keys can only be toggled when French is the main language.

```
  \define@key{FB}{StandardLayout}[true]%
          {\ifFB@mainlanguage@FR
             \csname FBStandardLayout#1\endcsname
           \else
             \PackageWarning{french.ldf}%
               {Option `StandardLayout' skipped:\MessageBreak
                French is *not* babel's last option.\MessageBreak
                Reported}%
           \fi
           \ifFBStandardLayout
             \FBStandardListSpacingtrue
             \FBStandardItemizeEnvtrue
             \FBStandardItemLabelstrue
             \FBStandardEnumerateEnvtrue
             \FBIndentFirstfalse
             \FBFrenchFootnotesfalse
             \FBAutoSpaceFootnotesfalse
           \else
             \FBStandardListSpacingfalse
             \FBStandardItemizeEnvfalse
             \FBStandardItemLabelsfalse
             \FBStandardEnumerateEnvfalse
             \FBIndentFirsttrue
             \FBFrenchFootnotestrue
             \FBAutoSpaceFootnotestrue
           \fi}%
  \define@key{FB}{GlobalLayoutFrench}[true]%
          {\ifFB@mainlanguage@FR
             \csname FBGlobalLayoutFrench#1\endcsname
           \else
             \PackageWarning{french.ldf}%
               {Option `GlobalLayoutFrench' skipped:\MessageBreak
                French is *not* babel's last option.\MessageBreak
                Reported}%
           \fi}%
```

If this key is set to true when French is the main language, nothing to do: all flags keep their default value. If this key is set to false, nothing to do either: \babel@save will do the job at every language's switch.

```
  \define@key{FB}{ReduceListSpacing}[true]%
          {\csname FBReduceListSpacing#1\endcsname
           \ifFBReduceListSpacing \FBStandardListSpacingfalse
           \else \FBStandardListSpacingtrue\fi
```

```
          }%
\define@key{FB}{StandardListSpacing}[true]%
         {\csname FBStandardListSpacing#1\endcsname}%
\define@key{FB}{ListOldLayout}[true]%
         {\csname FBListOldLayout#1\endcsname
          \ifFBListOldLayout
            \FBStandardEnumerateEnvtrue
            \renewcommand*{\FrenchLabelItem}{\textendash}%
          \fi}%
\define@key{FB}{CompactItemize}[true]%
         {\csname FBCompactItemize#1\endcsname
          \ifFBCompactItemize
            \FBStandardItemizeEnvfalse
            \FBStandardEnumerateEnvfalse
          \else
            \FBStandardItemizeEnvtrue
            \FBStandardEnumerateEnvtrue
          \fi}%
\define@key{FB}{StandardItemizeEnv}[true]%
         {\csname FBStandardItemizeEnv#1\endcsname}%
\define@key{FB}{StandardEnumerateEnv}[true]%
         {\csname FBStandardEnumerateEnv#1\endcsname}%
\define@key{FB}{StandardItemLabels}[true]%
         {\csname FBStandardItemLabels#1\endcsname}%
\define@key{FB}{ItemLabels}%
         {\renewcommand*{\FrenchLabelItem}{#1}}%
\define@key{FB}{ItemLabeli}%
         {\renewcommand*{\Frlabelitemi}{#1}}%
\define@key{FB}{ItemLabelii}%
         {\renewcommand*{\Frlabelitemii}{#1}}%
\define@key{FB}{ItemLabeliii}%
         {\renewcommand*{\Frlabelitemiii}{#1}}%
\define@key{FB}{ItemLabeliv}%
         {\renewcommand*{\Frlabelitemiv}{#1}}%
\define@key{FB}{StandardLists}[true]%
         {\csname FBStandardLists#1\endcsname
          \ifFBStandardLists
            \FBStandardListSpacingtrue
            \FBStandardItemizeEnvtrue
            \FBStandardEnumerateEnvtrue
            \FBStandardItemLabelstrue
          \else
            \FBStandardListSpacingfalse
            \FBStandardItemizeEnvfalse
            \FBStandardEnumerateEnvfalse
            \FBStandardItemLabelsfalse
          \fi}%
\define@key{FB}{ListItemsAsPar}[true]%
         {\csname FBListItemsAsPar#1\endcsname}
\define@key{FB}{IndentFirst}[true]%
         {\csname FBIndentFirst#1\endcsname}%
\define@key{FB}{FrenchFootnotes}[true]%
         {\csname FBFrenchFootnotes#1\endcsname}%
```

```
\define@key{FB}{AutoSpaceFootnotes}[true]%
        {\csname FBAutoSpaceFootnotes#1\endcsname}%
\define@key{FB}{AutoSpacePunctuation}[true]%
        {\csname FBAutoSpacePunctuation#1\endcsname}%
\define@key{FB}{OriginalTypewriter}[true]%
        {\csname FBOriginalTypewriter#1\endcsname}%
\define@key{FB}{ThinColonSpace}[true]%
        {\csname FBThinColonSpace#1\endcsname
         \ifFBThinColonSpace
           \renewcommand*{\FBcolonspace}{\FBthinspace}%
         \fi}%
\define@key{FB}{ThinSpaceInFrenchNumbers}[true]%
        {\csname FBThinSpaceInFrenchNumbers#1\endcsname}%
\define@key{FB}{FrenchSuperscripts}[true]%
        {\csname FBFrenchSuperscripts#1\endcsname}
\define@key{FB}{LowercaseSuperscripts}[true]%
        {\csname FBLowercaseSuperscripts#1\endcsname}
\define@key{FB}{PartNameFull}[true]%
        {\csname FBPartNameFull#1\endcsname}%
\define@key{FB}{CustomiseFigTabCaptions}[true]%
        {\csname FBCustomiseFigTabCaptions#1\endcsname}%
\define@key{FB}{OldFigTabCaptions}[true]%
        {\csname FBOldFigTabCaptions#1\endcsname
         \ifFBOldFigTabCaptions
           \def\FB@capsep@fix{\babel@save\FBCaption@Separator
                  \def\FBCaption@Separator{\CaptionSeparator}}%
           \addto\extrasfrench{\FB@capsep@fix}%
           \ifdefined\extrasacadian
             \addto\extrasacadian{\FB@capsep@fix}%
           \fi
         \fi}%
\define@key{FB}{SmallCapsFigTabCaptions}[true]%
        {\csname FBSmallCapsFigTabCaptions#1\endcsname
         \ifFBSmallCapsFigTabCaptions
         \else \let\FBfigtabshape\relax \fi}%
\define@key{FB}{SuppressWarning}[true]%
        {\csname FBSuppressWarning#1\endcsname
         \ifFBSuppressWarning
           \renewcommand{\FBWarning}[1]{}%
         \fi}%
```

Here are the options controlling French guillemets spacing and the output of `\frquote{}`.

```
\define@key{FB}{INGuillSpace}[true]%
        {\csname FBINGuillSpace#1\endcsname
         \ifFBINGuillSpace
           \renewcommand*{\FBguillspace}{\space}%
         \fi}%
\define@key{FB}{InnerGuillSingle}[true]%
        {\csname FBInnerGuillSingle#1\endcsname}%
\define@key{FB}{EveryParGuill}[open]%
        {\expandafter\let\expandafter
           \FBeveryparguill\csname FBguill#1\endcsname
         \ifx\FBeveryparguill\FBguillopen
```

```
          \else\ifx\FBeveryparguill\FBguillclose
               \else\ifx\FBeveryparguill\FBguillnone
                    \else
                      \let\FBeveryparguill\FBguillopen
                      \FBWarning{Wrong value for `EveryParGuill':
                                 try `open',\MessageBreak
                                 `close' or `none'. Reported}%
                    \fi
               \fi
          \fi}%
  \define@key{FB}{EveryLineGuill}[open]%
         {\ifFB@luatex@punct
            \expandafter\let\expandafter
              \FBeverylineguill\csname FBguill#1\endcsname
            \ifx\FBeverylineguill\FBguillopen
            \else\ifx\FBeverylineguill\FBguillclose
                 \else\ifx\FBeverylineguill\FBguillnone
                      \else
                        \let\FBeverylineguill\FBguillnone
                        \FBWarning{Wrong value for `EveryLineGuill':
                                   try `open',\MessageBreak
                                   `close' or `none'. Reported}%
                      \fi
                 \fi
            \fi
          \else
            \FBWarning{Option `EveryLineGuill' skipped:%
                       \MessageBreak this option is for
                       LuaTeX *only*.\MessageBreak Reported}%
          \fi}%
```

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.

```
  \define@key{FB}{UnicodeNoBreakSpaces}[true]%
         {\ifFB@luatex@punct
            \csname FBucsNBSP#1\endcsname
            \ifFBucsNBSP \FB@ucsNBSP=\@ne \fi
          \else
            \FBWarning{Option `UnicodeNoBreakSpaces' skipped:%
                       \MessageBreak this option is for
                       LuaTeX *only*.\MessageBreak Reported}%
          \fi
         }%
```

Inputing 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@addGUILspace 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.

```
\def\FB@parse#1#2\endparse{\def\FB@second{#2}}%
\define@key{FB}{og}%
         {\ifFBunicode
```

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

```
            \ifFB@luatex@punct
              \FB@addGUILspace=1 \relax
            \fi
```

then with XeTeX it is a bit more tricky:

```
            \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).

```
              \XeTeXcharclass"13   = \FB@guilo
              \XeTeXcharclass"AB   = \FB@guilo
              \XeTeXcharclass"A0   = \FB@guilnul
              \XeTeXcharclass"202F = \FB@guilnul
            \fi
```

Issue a warning with older Unicode engines requiring active characters.

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

This is for conventional TeX engines:

```
            \newcommand*{\FB@@og}{%
               \ifFBfrench
                 \ifFB@spacing\FB@og\ignorespaces
                 \else\guillemotleft
                 \fi
               \else\guillemotleft\fi}%
            \AtBeginDocument{%
               \ifdefined\uc@dclc
```

Package inputenc with utf8x (ucs) encoding loaded, use \uc@dclc:

```
                 \uc@dclc{171}{default}{\FB@@og}%
               \else
```

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

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

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

```
                   \ifdefined\mule@def
                     \mule@def{11}{\FB@@og}%
```

```
                    \else
                      \ifdefined\DeclareInputText
                        \@tempcnta`#1\relax
                        \DeclareInputText{\the\@tempcnta}{\FB@@og}%
                      \else
```

Package inputenc not loaded, no way...

```
                        \FBWarning{Option `og' requires package
                                   inputenc;\MessageBreak reported}%
                      \fi
                    \fi
                  \else
```

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

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

Same code for the closing quote.

```
  \define@key{FB}{fg}%
          {\ifFBunicode
             \ifFB@luatex@punct
               \FB@addGUILspace=1 \relax
             \fi
             \ifFB@xetex@punct
               \XeTeXcharclass"14   = \FB@guilf
               \XeTeXcharclass"BB   = \FB@guilf
               \XeTeXcharclass"A0   = \FB@guilnul
               \XeTeXcharclass"202F = \FB@guilnul
             \fi
             \ifFB@active@punct
               \FBWarning{Option fg=» not supported with this version
                          of\MessageBreak LuaTeX/XeTeX; reported}%
             \fi
           \else
             \newcommand*{\FB@@fg}{%
                \ifFBfrench
                  \ifFB@spacing\FB@fg
                  \else\guillemotright
                  \fi
                \else\guillemotright\fi}%
             \AtBeginDocument{%
                \ifdefined\uc@dclc
                  \uc@dclc{187}{default}{\FB@@fg}%
                \else
                  \FB@parse#1\endparse
                  \ifx\FB@second\@empty
                    \ifdefined\mule@def
                      \mule@def{27}{{\FB@@fg}}%
                    \else
                      \ifdefined\DeclareInputText
                        \@tempcnta`#1\relax
                        \DeclareInputText{\the\@tempcnta}{\FB@@fg}%
```

```
                    \else
                      \FBWarning{Option `fg' requires package
                                 inputenc;\MessageBreak reported}%
                    \fi
                  \fi
                \else
                  \DeclareUnicodeCharacter{00BB}{\FB@@fg}%
                \fi
              \fi}%
          \fi
         }%
}
```

`\FBprocess@options` `\FBprocess@options` will be executed at `\begin{document}`: it first checks about packages loaded in the preamble (possibly after Babel) which customise lists: currently `enumitem`, `paralist` and `enumerate`; then it processes the options as set by `\frenchsetup{}` or forced for compatibility with packages loaded in the preamble. When French is the main language, `\extrasfrench` and `\captionsfrench` *have already been processed* by Babel at `\begin{document}` *before* `\FBprocess@options`.

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

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

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

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

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

```
\def\FB@ufl{\update@frenchlists}
\ifFB@mainlanguage@FR
  \update@frenchlists
\else
  \ifFBStandardItemizeEnv
  \else
    \PackageWarning{french.ldf}%
      {babel-french will not customize lists' layout\MessageBreak
       when French is not the main language,\MessageBreak
       reported}%
  \fi
\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.

```
\ifFBAutoSpacePunctuation
  \autospace@beforeFDP
\else
  \noautospace@beforeFDP
\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.

```
\ifFBOriginalTypewriter
\else
  \let\ttfamilyORI\ttfamily
  \let\rmfamilyORI\rmfamily
  \let\sffamilyORI\sffamily
  \let\ttfamily\ttfamilyFB
  \let\rmfamily\rmfamilyFB
  \let\sffamily\sffamilyFB
\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.

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

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

```
\ifFBFrenchSuperscripts
  \DeclareRobustCommand*{\up}{%
    \texorpdfstring{\@ifstar{\FB@up@fake}{\fup}}{}%
    }
\else
  \DeclareRobustCommand*{\up}{%
    \texorpdfstring{\@ifstar{\FB@up@fake}{\textsuperscript}}{}%
    }
\fi
```

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

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

```
\ifFB@koma
   \ifx\captionformat\FB@std@capsep
     \ifFBCustomiseFigTabCaptions
        \renewcommand*{\captionformat}{\CaptionSeparator}%
     \else
        \renewcommand*{\captionformat}{{\autospace@beforeFDP :\ }}%
     \fi
   \fi
\fi
\@ifclassloaded{memoir}%
   {\ifx\@contdelim\FB@std@capsep
      \ifFBCustomiseFigTabCaptions
        \captiondelim{\CaptionSeparator}%
      \else
        \captiondelim{{\autospace@beforeFDP : }}%
      \fi
```

```
     \fi}{}%
  \@ifclassloaded{beamer}%
     {\protected@edef\FB@capsep{%
         \csname beamer@@tmpl@caption label separator\endcsname}%
      \ifx\FB@capsep\FB@std@capsep
        \ifFBCustomiseFigTabCaptions
           \defbeamertemplate{caption label separator}{FBcustom}{%
               \CaptionSeparator}%
           \setbeamertemplate{caption label separator}[FBcustom]%
        \else
           \defbeamertemplate{caption label separator}{FBcolon}{%
               {\autospace@beforeFDP : }}%
           \setbeamertemplate{caption label separator}[FBcolon]%
        \fi
      \fi}{}%
```

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

```
  \ifFBShowOptions
    \GenericWarning{* }{%
     *** List of possible options for babel-french ***\MessageBreak
     [Default values between brackets when french is loaded *LAST*]%
     \MessageBreak
     ShowOptions [false]\MessageBreak
     StandardLayout [false]\MessageBreak
     GlobalLayoutFrench [true]\MessageBreak
     PartNameFull [true]\MessageBreak
     IndentFirst [true]\MessageBreak
     ListItemsAsPar [false]\MessageBreak
     StandardListSpacing [false]\MessageBreak
     StandardItemizeEnv [false]\MessageBreak
     StandardEnumerateEnv [false]\MessageBreak
     StandardItemLabels [false]\MessageBreak
     ItemLabels=\textemdash, \textbullet,
        \protect\ding{43},... [\textendash]\MessageBreak
     ItemLabeli=\textemdash, \textbullet,
        \protect\ding{43},... [\textendash]\MessageBreak
     ItemLabelii=\textemdash, \textbullet,
        \protect\ding{43},... [\textendash]\MessageBreak
     ItemLabeliii=\textemdash, \textbullet,
        \protect\ding{43},... [\textendash]\MessageBreak
     ItemLabeliv=\textemdash, \textbullet,
        \protect\ding{43},... [\textendash]\MessageBreak
     StandardLists [false]\MessageBreak
     ListOldLayout [false]\MessageBreak
     FrenchFootnotes [true]\MessageBreak
     AutoSpaceFootnotes [true]\MessageBreak
     AutoSpacePunctuation [true]\MessageBreak
     ThinColonSpace [false]\MessageBreak
     OriginalTypewriter [false]\MessageBreak
     UnicodeNoBreakSpaces [false]\MessageBreak
     og= <left quote character>, fg= <right quote character>%
     INGuillSpace [false]\MessageBreak
     EveryParGuill=open, close, none [open]\MessageBreak
     EveryLineGuill=open, close, none
```

```
                  [open in LuaTeX, none otherwise]\MessageBreak
    InnerGuillSingle [false]\MessageBreak
    ThinSpaceInFrenchNumbers [false]\MessageBreak
    SmallCapsFigTabCaptions [true]\MessageBreak
    CustomiseFigTabCaptions [true]\MessageBreak
    OldFigTabCaptions [false]\MessageBreak
    FrenchSuperscripts [true]\MessageBreak
    LowercaseSuperscripts [true]\MessageBreak
    SuppressWarning [false]\MessageBreak
    \MessageBreak
    ****************************************%
    \MessageBreak\protect\frenchsetup{ShowOptions}}
  \fi
}
```

At `\begin{document}`, we have to provide an `\xspace` command in case the `xspace` package is not loaded, do some setup for `hyperref`'s bookmarks, execute `\FBprocess@options`, switch LuaTeX punctuation on and issue some warnings if necessary.

```
\AtBeginDocument{%
  \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`.

```
  \FBprocess@options
```

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

```
  \ifFBucsNBSP
    \renewcommand*{\FBmedkern}{\char"202F\relax}%
    \renewcommand*{\FBthickkern}{\char"A0\relax}%
    \ifFBThinSpaceInFrenchNumbers
      \renewcommand*{\FBthousandsep}{\char"202F\relax}%
    \else
      \renewcommand*{\FBthousandsep}{\char"A0\relax}%
    \fi
  \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*`!

```
  \begingroup
    \newcommand{\FBOTone}{OT1}%
    \ifx\encodingdefault\FBOTone
      \FBWarning{OT1 encoding should not be used for French.%
                 \MessageBreak
                 Add \protect\usepackage[T1]{fontenc} to the
                 preamble\MessageBreak of your document; reported}%
    \fi
  \endgroup
}
```

2.12 French lists

\listFB
\listORI
\FB@listVsettings

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

```
\let\listORI\list
\let\endlistORI\endlist
\newdimen\FB@parskip
\def\FB@listVsettings{%
      \setlength{\topsep}{0.8ex plus 0.4ex minus 0.4ex}%
      \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.

```
      \FB@parskip=\parskip
      \addtolength{\topsep}{-\FB@parskip}%
      \addtolength{\partopsep}{\FB@parskip}%
      \setlength{\itemsep}{0.4ex plus 0.2ex minus 0.2ex}%
      \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 preceeded by the same vertical space as paragraphs located outside lists; the vertical skip before items (\itemsep + \parsep) doesn't need to be enlarged.

```
      \ifdim\FB@parskip>0pt
        \setlength{\parsep}{\FB@parskip}%
        \addtolength{\itemsep}{-\FB@parskip}%
      \fi
}
\def\listFB#1#2{\listORI{#1}{\FB@listVsettings #2}}
\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. 5.

\FrenchLabelItem
\Frlabelitemi
\Frlabelitemii
\Frlabelitemiii
\Frlabelitemiv

Default labels for French itemize-lists (same label for all levels):

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

\listindentFB
\descindentFB
\labelindentFB
\labelwidthFB

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

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

\leftmarginFB
\FB@listHsettings

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

```
\newdimen\leftmarginFB
\def\FB@listHsettings{%
  \ifFBListItemsAsPar
```

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

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

Default layout: labels hanging into the list left margin.

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

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

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

\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).

```
\def\FB@itemizesettings{%
    \ifFBStandardListSpacing
    \else
      \setlength{\topsep}{\z@}%
      \setlength{\partopsep}{\z@}%
      \FB@parskip=\parskip
      \addtolength{\topsep}{-\FB@parskip}%
      \addtolength{\partopsep}{\FB@parskip}%
      \setlength{\itemsep}{\z@}%
      \setlength{\parsep}{\z@}%
      \ifdim\FB@parskip>0pt
        \setlength{\parsep}{\FB@parskip}%
        \addtolength{\itemsep}{-\FB@parskip}%
      \fi
    \fi
    \settowidth{\labelwidth}{\csname\@itemitem\endcsname}%
    \ifFBListOldLayout
      \setlength{\leftmargin}{\labelwidth}%
      \addtolength{\leftmargin}{\labelsep}%
      \addtolength{\leftmargin}{\parindent}%
    \else
      \FB@listHsettings
    \fi
}
```

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

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

\def\setlabelitemsFB{%
  \let\labelitemi\Frlabelitemi
  \let\labelitemii\Frlabelitemii
  \let\labelitemiii\Frlabelitemiii
  \let\labelitemiv\Frlabelitemiv
  \ifdim\labelwidthFB<\z@
    \settowidth{\labelwidthFB}{\FrenchLabelItem}%
  \fi
}
```

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

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

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

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

```
\def\descriptionFB{%
      \list{}{\FB@listHsettings
              \labelwidth=\z@
```

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

`\update@frenchlists` `\bbl@frenchlistlayout`

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

```
\def\update@frenchlists{%
  \setlistindentFB
  \ifFBStandardListSpacing
  \else \let\list\listFB \fi
  \ifFBStandardItemizeEnv
  \else \let\itemize\itemizeFB \fi
  \ifFBStandardItemLabels
  \else \setlabelitemsFB \fi
  \ifFBStandardEnumerateEnv
  \else \let\enumerate\enumerateFB \let\description\descriptionFB \fi
}
```

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@ufl` as `\relax`, it will be redefined later 'AtBeginDocument' by `\FBprocess@options` as `\update@frenchlists`, see p. 63.

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

```
\def\FB@ufl{\relax}
\def\bbl@frenchlistlayout{%
  \ifFBGlobalLayoutFrench
  \else
    \babel@save\list         \babel@save\itemize
    \babel@save\enumerate    \babel@save\description
    \babel@save\labelitemi   \babel@save\labelitemii
    \babel@save\labelitemiii \babel@save\labelitemiv
    \FB@ufl
```

```
  \fi
}
\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.

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

2.14 Formatting footnotes

The `bigfoot` package deeply changes the way footnotes are handled. When `bigfoot` is loaded, we just warn the user that `babel-french` will drop the customisation of 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!).
We 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 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`.

```
\AtBeginDocument{\@ifpackageloaded{bigfoot}%
                    {\PackageInfo{french.ldf}%
                      {bigfoot package in use.\MessageBreak
                       babel-french will NOT customise footnotes;%
                       \MessageBreak reported}}%
                    {\let\@footnotemarkORI\@footnotemark
                     \def\@footnotemarkFB{\leavevmode\unskip\unkern
                                          \,\@footnotemarkORI}%
                     \ifFBAutoSpaceFootnotes
                       \let\@footnotemark\@footnotemarkFB
                     \fi}%
                }
```

\@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).

```
\newdimen\parindentFFN
\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.

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

\@makefntextFB's definition is now tuned according to the document's class for better compatibility.
Koma-script classes provide \deffootnote, a handy command to customise the footnotes' layout (see English manual scrguien.pdf); it redefines \@makefntext and \@@makefnmark. First, save the original definitions.

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

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

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

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

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

Restore the original definitions.

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

Definitions for the memoir class:

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

(see original definition in memman.pdf)

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

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

Definitions for the beamer class:

```
\@ifclassloaded{beamer}
```

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

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

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

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

The rest of \@makefntext's customisation is done at the \begin{document}. We save the original definition of \@makefntext, and then redefine \@makefntext according to the value of flag \ifFBFrenchFootnotes (true or false). Koma-script classes require a special treatment.
The LuaTeX command \localleftbox and \FBeverypar@quote used by \frquote{} have to be reset inside footnotes; done for LaTeX based formats only.

```
\providecommand\localleftbox[1]{}
\AtBeginDocument{%
   \@ifpackageloaded{bigfoot}{}%
      {\ifdim\parindentFFN<10in
       \else
         \parindentFFN=\parindent
         \ifdim\parindentFFN<1.5em \parindentFFN=1.5em \fi
       \fi
       \settowidth{\FBfnindent}{\dotFFN\kernFFN}%
       \addtolength{\FBfnindent}{\parindentFFN}%
       \let\@makefntextORI\@makefntext
       \ifFB@koma
```

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

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

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

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

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

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

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

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

```
      \fi
    }%
}
```

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.

```
\newcommand*{\AddThinSpaceBeforeFootnotes}{\FBAutoSpaceFootnotestrue}
\newcommand*{\FrenchFootnotes}{\FBFrenchFootnotestrue}
\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.

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

```
<*acadian>
\PackageInfo{acadian.ldf}%
  {`acadian' dialect is currently\MessageBreak
   *absolutely identical* to the\MessageBreak
   `french' language; reported}
</acadian>
<*canadien>
\PackageWarning{canadien.ldf}%
  {Option `canadien' for Babel is *deprecated*,\MessageBreak
   it might be removed sooner or later.  Please\MessageBreak
   use `acadian' instead; reported}%
\def\CurrentOption{acadian}

\def\datecanadien{\dateacadian}
\def\captionscanadien{\captionsacadian}
\def\extrascanadien{\extrasacadian}
\def\noextrascanadien{\noextrasacadian}
</canadien>
<*francais>
\PackageWarning{francais.ldf}%
```

```
  {Option `francais' for Babel is *deprecated*,\MessageBreak
   it might be removed sooner or later.  Please\MessageBreak
   use `french' instead; reported}%
\chardef\l@francais\l@french
\def\CurrentOption{french}
</francais>
```

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

```
<*frenchb>
\def\bbl@tempa{frenchb}
\ifx\CurrentOption\bbl@tempa
  \chardef\l@frenchb\l@french
  \def\CurrentOption{french}
  \PackageWarning{babel-french}%
    {Option `frenchb' for Babel is *deprecated*,\MessageBreak
     it might be removed sooner or later.  Please\MessageBreak
     use `french' instead; reported}
\else
  \def\bbl@tempa{francais}
  \ifx\CurrentOption\bbl@tempa
    \chardef\l@francais\l@french
    \def\CurrentOption{french}
```

Plain formats: no warning when `francais.sty` loads `frenchb.ldf` (Babel pre-3.13).

```
    \ifx\magnification\@undefined
      \PackageWarning{babel-french}%
        {Option `francais' for Babel is *deprecated*,\MessageBreak
         it might be removed sooner or later.  Please\MessageBreak
         use `french' instead; reported}
    \fi
  \else
    \def\bbl@tempa{canadien}
    \ifx\CurrentOption\bbl@tempa
      \def\CurrentOption{acadian}
      \PackageWarning{babel-french}%
        {Option `canadien' for Babel is *deprecated*,\MessageBreak
         it might be removed sooner or later.  Please\MessageBreak
         use `acadian' instead; reported}
    \fi
  \fi
\fi
</frenchb>
<acadian|canadien|frenchb|francais>\input french.ldf\relax
<acadian|canadien>\let\extrasacadian\extrasfrench
<acadian|canadien>\let\noextrasacadian\noextrasfrench
<acadian|canadien|frenchb|francais|french>\endinput
```

3 Change History

Changes are listed in reverse order (latest first) and limited to babel-french v3.