A Babel language definition file for French

frenchb.dtx v3.3a, 2017/04/30

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 and Denis Bitouzé. Thanks to all of them!
LaTeX-2.09 is no longer supported. This new version (3.x) has been designed to be used only with LaTeX 2$_\varepsilon$ and Plain formats based on TeX, pdfTeX, LuaTeX or XeTeX engines.
Changes between version 3.0 and v3.3a are listed in subsection 1.4 p. 10.
An extensive documentation is available in French here:
http://daniel.flipo.free.fr/frenchb

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.
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*[2]:

1. the first paragraph of each section is indented (LaTeX only);
2. the default items in itemize environment are set to '—' instead of '•', and all vertical spacing and glue is deleted; it is possible to change '—' to something else ('–' for instance) using `\frenchsetup{}` (see section 1.2 p. 5);
3. vertical spacing in general LaTeX lists is shortened;
4. footnotes are displayed "à la française".
5. the separator following the table or figure number in captions is printed as ' – ' instead of ': '; for changing this see 1.2.2 p. 9.

[1]The file described in this section has version number v3.3a and was last revised on 2017/04/30.
[2] 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. 5).

Regarding local typography, the command `\selectlanguage{french}` switches to the French language[3], with the following effects:

1. French hyphenation patterns are made active;
2. 'high punctuation' characters (`: ; ! ?`) automatically add correct spacing in French; this is achieved using callbacks in Lua(La)TeX 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 commands `\og` and `\fg` which work in LaTeX 2$_\varepsilon$and PlainTeX, their appearance depending on what is available to draw them; even if you use LaTeX 2$_\varepsilon$ *and* T1-encoding, you should refrain from entering them as `<<~French quotation~>>`: `\og` and `\fg` provide better horizontal spacing (controlled by `\FBguillspace`). If French quote characters are available on your keyboard, you can use them, to get proper spacing in LaTeX 2$_\varepsilon$ see option `og=«, fg=»` p. 8.

   `\og` and `\fg` can be used outside French, they typeset then English quotes " and ".

   A new command `\frquote{}` has been added in version 3.1 to enter French quotations. `\frquote{`*texte*`}` is equivalent to `\og` *texte* `\fg{}` for short quotations. 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.

   `\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.

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

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

2. 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.

3. Family names should be typeset in small capitals and never be hyphenated, the macro `\bsc` (boxed small caps) does this, e.g., `L.~\bsc{Lamport}` will print the same as `L.~\mbox{\textsc{Lamport}}`. Note that composed names (such as Dupont-Durant) may now be hyphenated on explicit hyphens, this differs from babel-french v. 1.x.

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

5. Abbreviations for “Numéro(s)” and “numéro(s)” (N^{o} N^{os} n^{o} and n^{os}) are obtained via the commands `\No`, `\Nos`, `\no`, `\nos`.

6. 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 an nobreak space), or for alcohols” strengths (e.g., “`45\degres`” with *no* space in French).

7. 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 space has to be added in lists and intervals: `$[0,\ 1]$`, `$(x,\ y)$`. `\StandardMathComma` switches back to the standard behaviour of the comma in French.

   The `icomma` package is an alternative workaround.

8. 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`, see `numprint.pdf` for more information.

9. 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}`

`\frenchsetup{}` and `\frenchbsetup{}` 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. This option can be used to avoid conflicts with classes or packages which customise lists or footnotes.

When French is not the main language, `StandardLayout=false` can be misused to ensure French typography (in French only). This is a *bad practice*: the document layout should not be altered by language switches.

`GlobalLayoutFrench=false (true∗)` should no longer be used; it was intended to emulate, when French is the main language, what prior versions of babel-french (pre-2.2) did: lists, and first paragraphs of sections would be displayed the standard way in other languages than French, and "à la française" in French. Note that the layout of footnotes is language independent anyway (see below `FrenchFootnotes` and `AutoSpaceFootnotes`).

`ReduceListSpacing=false (true∗)` ; babel-french reduces the values of the vertical spaces used in the *all* list environments in French (this includes itemize, enumerate, description, but also abstract, quote, quotation and verse and possibly others). Setting this option to `false` reverts to the standard settings of the `list` environment.

`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.

`CompactItemize=false (true∗)` ; should no longer be used (kept only for backward compatibility), it is replaced by the next two options.

`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 `false` reverts to the standard definition of `itemize`.

`StandardEnumerateEnv=true (false*)` ; starting with version 2.6 babel-french redefines the `enumerate` and `description` environments to make left margins match those of the French version of `itemize` lists. Setting this option to `false` 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 the example `\ding{43}` requires `\usepackage{pifont}`.

`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 `ReduceListSpacing=false`, `StandardItemizeEnv=true`, `StandardEnumerateEnv=true` and `StandardItemLabels=true`.

`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.

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

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

`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.

`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 nobreak 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), 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 [4], 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. 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 10:55}`).

`ThinColonSpace=true (false)` changes the inter-word unbreakable 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.

`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).

`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".

`CustomiseFigTabCaptions=false (true*)` ; when `false` the default separator (colon) is used instead of `\CaptionSeparator`. Anyway, babel-french makes sure that the colon will be typeset with proper preceeding space in French.

`OldFigTabCaptions=true (false)` is to be used 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.

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

`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).

`SuppressWarning=true (false)` ; can be turned to `true` if you are bored with babel-french’s warnings.

`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 lesser 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.

`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 `\og` and `\fg`. 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`» (with or without spaces) to get properly typeset French quotes. This option works with LuaLaTeX and XeLaTeX; with pdfLaTeX it requires `inputenc` to be loaded with a proper encoding: 8-bits encoding (latin1, latin9, ansinew, applemac,. . .) or multi-byte encoding (utf8, utf8x).

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 Captions

Caption names can be customised in French using the simplified syntax introduced by babel 3.9, for instance: `\def\frenchproofname{Preuve}`. The older syntax `\addto\captionsfrench{\def\proofname{Preuve}}` still works. Keep in mind that *only* `french` can be used to redefine captions, even if babel's option was entered as `francais` or `frenchb`.
When French is the main language, by default (see below) babel-french changes 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}{...}`.
When French is not the main language, the colon is preserved for all languages but babel-french makes sure that a proper space is typeset before it.
Three new options are provided: if `CustomiseFigTabCaptions` is set to `false` the colon will be used as separator in all languages, with a proper space before the colon in French. 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 LaTeX 2_ε I suggest this:

- run `pdfLaTeX` on the following file, with the encoding suitable for your machine (*my-encoding* will be `latin1` for Unix machines, `ansinew` for PCs running Windows, `applemac` or `latin1` for Macintoshs, or `utf8`...

  ```
  %%% Test file for French hyphenation.
  \documentclass{article}
  \usepackage[my-encoding]{inputenc}
  \usepackage[T1]{fontenc} % Use LM fonts
  \usepackage{lmodern}     % for French
  \usepackage[frenchb]{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 3.3?

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.
Tne 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' [5]. 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.

  Please note that this code, still experimental, is likely to change until LuaTeX itself has reached version 1.0.

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.

[5] The current babel-french version requires LuaTeX v. 0.95 as included in TL2016, see above.

2 The code

2.1 Initial setup

If `frenchb.ldf` was loaded with babel's options `francais` or `frenchb`, we make it behave as if `french` was specified. In Plain formats, `@` catcode is not 'letter'.

```
\chardef\atcatcode=\catcode‘\@
\catcode‘\@=11\relax
\def\bbl@tempa{francais}
\ifx\CurrentOption\bbl@tempa
  \let\l@francais\l@french
  \def\captionsfrancais{\captionsfrench}
  \def\datefrancais{\datefrench}
  \def\extrasfrancais{\extrasfrench}
  \def\noextrasfrancais{\extrasfrench}
  \def\CurrentOption{french}
\fi
\def\bbl@tempa{frenchb}
\ifx\CurrentOption\bbl@tempa
  \let\l@frenchb\l@french
  \def\captionsfrenchb{\captionsfrench}
  \def\datefrenchb{\datefrench}
  \def\extrasfrenchb{\extrasfrench}
  \def\noextrasfrenchb{\extrasfrench}
  \def\CurrentOption{french}
\fi
\catcode‘\@=\atcatcode \let\atcatcode\relax
```

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

```
\LdfInit\CurrentOption\captionsfrench
```

Make sure that `\l@french` is defined (possibly as 0). `babel.def` now (3.9i) defines `\l@<languagename>` also for eTeX, LuaTeX and XeTeX formats which set `\lang@<languagename>`.

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

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

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

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

Quit if babel’s version is less than 3.9i.

```
\let\bbl@tempa\relax
\ifx\babeltags\@undefined
   \let\bbl@tempa\endinput
   \ifLaTeXe
      \PackageError{frenchb.ldf}
         {frenchb requires babel v.3.9i.\MessageBreak
          Aborting here}
         {Please upgrade Babel!}
   \else
      \fb@error{frenchb requires babel v.3.9i.\\
                Aborting here}
               {Please upgrade Babel!}
   \fi
\fi
```

```
\bbl@tempa
```

`frenchb.ldf` can be loaded with options canadien or acadian, which both stand for Canadian French. Internally, acadian will be the name of the corresponding babel’s dialect, so we set `\CurrentOption` to acadian in both cases. If no specific hyphenation patterns are available, Canadian French will use the French ones.
TODO: Canadian French hyphenation doesn’t work with LuaTeX.

```
\ifx\l@acadian\@undefined
   \ifx\l@canadien\@undefined
      \adddialect\l@acadian\l@french
      \adddialect\l@canadien\l@french
   \else
      \adddialect\l@acadian\l@canadien
   \fi
\else
  \adddialect\l@canadien\l@acadian
\fi
\def\bbl@tempa{canadien}
\ifx\CurrentOption\bbl@tempa
  \def\captionscanadien{\captionsacadian}
  \def\datecanadien{\dateacadian}
  \def\extrascanadien{\extrasacadian}
  \def\noextrascanadien{\extrasacadian}
  \def\CurrentOption{acadian}
\fi
```

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

```
\expandafter\providehyphenmins\expandafter{\CurrentOption}{\tw@\thr@@}
```

`\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.
We cannot rely on ε-TeX’s `\ifdefined` at this stage, as it is not defined in Plain TeX format.

```
\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
```

`\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” 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.

```
\@namedef{extras\CurrentOption}{%
     \babel@savevariable{\lccode`\'}%
     \ifFBunicode
        \babel@savevariable{\lccode"2019}%
        \lccode`\'="2019\lccode"2019="2019
     \else
        \lccode`\'=`\'
     \fi
}
\@namedef{noextras\CurrentOption}{}
```

Let’s define a handy command for adding stuff to `\extras\CurrentOption`, `\noextras\CurrentOption` or `\captions\CurrentOption` but first let’s save the value of `\CurrentOption` for later use in `\frenchsetup{}` (‘AfterEndOfPackage’, `\CurrentOption` will be lost).

```
\let\FB@CurOpt\CurrentOption
\newcommand*{\FB@addto}[2]{%
    \expandafter\addto\csname #1\FB@CurOpt\endcsname{#2}}
```

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

```
\FB@addto{extras}{\bbl@frenchspacing}
\FB@addto{noextras}{\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`

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

`\ifFB@luatex@punct` Three internal flags are needed for the three different techniques used for ‘high punctuation’ management.
With LuaTeX, starting with version 0.95, 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<95
    \ifx\PackageWarning\@undefined
      \fb@warning{Please upgrade LuaTeX to version 0.95 or above!\\%
         frenchb will make high punctuation characters (;:!?) active\\%
         with LuaTeX < 0.95.}%
    \else
      \PackageWarning{frenchb.ldf}{Please upgrade LuaTeX
         to version 0.95 or above!\MessageBreak
         frenchb will make high punctuation characters\MessageBreak
         (;:!?) active with LuaTeX < 0.95;\MessageBreak reported}%
    \fi
  \else
    \FB@luatex@puncttrue\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 4095 instead of 255.

```
\newcount\FB@nonchar
\newif\ifFB@xetex@punct
\begingroup\expandafter\expandafter\expandafter\endgroup
\expandafter\ifx\csname XeTeXinterchartokenstate\endcsname\relax
\else
  \FB@xetex@puncttrue\FB@active@punctfalse
  \ifdim\the\XeTeXversion\XeTeXrevision pt<0.99994pt
    \FB@nonchar=255 \relax
  \else
    \FB@nonchar=4095 \relax
  \fi
\fi
```

`\FBcolonspace` `\FBthinspace` 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, both are user customisable in the preamble.

```
\newcommand*{\FBcolonspace}{\space}
\newcommand*{\FBthinspace}{\hskip.5\fontdimen2\font \relax}
```

These commands will be converted into toks 'AtBeginDocument' for LuaTeX.

```
\newtoks\FBcolonsp
\newtoks\FBthinsp
```

With LuaTeX and XeTeX engines, babel-french handles French quotes together with 'high punctuation'; 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).

```
\newcommand*{\FB@spacing@on}{%
  \ifFB@luatex@punct
    \FB@spacing=1 \relax
  \else
    \FB@spacingtrue
  \fi}
\newcommand*{\FB@spacing@off}{%
  \ifFB@luatex@punct
    \FB@spacing=0 \relax
  \else
    \FB@spacingfalse
  \fi}
```

2.2.1 Punctuation with LuaTeX

The following part holds specific code for punctuation with modern LuaTeX engines, i.e. version 0.95 (included in TL2016) or newer.
We define three LuaTeX attributes to control spacing in French for 'high punctuation' and quotes, making sure that `\newattribute` is defined.

```
\ifFB@luatex@punct
  \begingroup\expandafter\expandafter\expandafter\endgroup
  \expandafter\ifx\csname newluafunction\endcsname\relax
```

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

```
    \input ltluatex.tex
  \fi
```

`\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 nobreak 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 '»'.

```
  \newattribute\FB@spacing      \FB@spacing=1 \relax
  \newattribute\FB@addDPspace   \FB@addDPspace=1 \relax
  \newattribute\FB@addGUILspace \FB@addGUILspace=0 \relax
  \ifLaTeXe
    \PackageInfo{frenchb.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
\fi
```

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

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

```
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 nobreak space U+00A0 or a nobreak thin space U+202F 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 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’ into toks `\FBthinsp`, `\FBcolonsp` and `\FBguillsp`; the latter are processed by the next function `get_glue` which returns a table of three values which are fractions of `\fontdimen2`, `\fontdimen3` and `\fontdimen4`.

```
local function get_glue(toks)
  local t = nil
  local f = string.match(toks, "\092hskip%s*([%d%.]*)%s*\092fontdimen")
  if f == "" then f = 1 end
  if f then
     t = {f, 0, 0}
     f = string.match(toks, "plus%s*([%d%.]*)%s*\092fontdimen")
     if f == "" then f = 1 end
     if f then
        t[2] = f
        f = string.match(toks, "minus%s*([%d%.]*)%s*\092fontdimen")
        if f == "" then f = 1 end
        if f then
           t[3] = f
        end
     end
  elseif string.match(toks, "\092F?B?thinspace") then
     t = {0.5, 0, 0}
  elseif string.match(toks, "\092space") then
     t = {1, 1, 1}
  end
  return t
end
local colngl = get_glue(tex.toks['FBcolonsp']) or {1, 1, 1}
local thingl = get_glue(tex.toks['FBthinsp'])  or {.5, 0, 0}
local guilgl = get_glue(tex.toks['FBguillsp']) or {.8, .3, .8}
```

The next function converts glue sizes returned in fontdimens by function `get_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 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 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`). Constant FR=`lang.id(french)` is defined by command `\activate@luatexpunct`.

```
local function french_punctuation (head)
  for item in node.traverse_id(GLYPH, head) do
    local lang = item.lang
    local char = item.char
    local fid  = item.font
    local FRspacing = has_attribute(item, FBspacing)
    FRspacing = FRspacing and FRspacing > 0
    local SIG  = has_attribute(item, addGUILspace)
    SIG = SIG and SIG >0
    if lang == FR and FRspacing and
                      FB_punct_left[char] and fid > 0 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 item is a glue, check its natural width, only positive glues (actually glues > 1 sp, for tabular ‘l’ columns) are to be replaced by a nobreakspace.

```
      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 one of these three conditions is met: a) the previous character is part of type `FB_punct_null` (this avoids spurious spaces in strings like `(!)` or `??`), b) a null glue (actually glues <= 1 sp for tabulars) preceeds the punctuation character, c) the punctuation character starts a paragraph or an `\hbox{}`.

```
      if FB_punct_thin[char] or FB_punct_thick[char] then
         local SBDP = has_attribute(item, addDPspace)
         local auto = SBDP and SBDP > 0
         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
         if FB_punct_thick[char] then
            fbglue = new_glue_scaled(fid,colngl)
         else
            fbglue = new_glue_scaled(fid,thingl)
         end
```

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
            insert_node_before(head, item, copy_node(nobreak))
            insert_node_before(head, item, copy_node(fbglue))
         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 fbglue = new_glue_scaled(fid,guilgl)
         if addgl and fbglue then
            if is_glue then
               head = remove_node(head,prev,true)
            end
            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*.

```
   if lang == FR and FRspacing and FB_punct_right[char]
                                 and fid > 0 and 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
         if next_id == GLYPH then
            next_char = next.char
```

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

```
         elseif 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 == GLYPH then
                     next_char = nextnext.char
                  end
               end
            end
         end
      end
      local is_glue = next_id == GLUE
      if is_glue then
         glue_wd = next.width
      end
      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 fbglue = new_glue_scaled(fid,guilgl)
      if addgl and fbglue then
         if is_glue then
            head = remove_node(head,next,true)
         end
         insert_node_after(head, item, copy_node(fbglue))
         insert_node_after(head, item, copy_node(nobreak))
      end
   end
  end
  return head
end
return french_punctuation
```

`\FB@luatex@punct@french` As a language tag is part of glyph nodes in LuaTeX, nothing needs to be added to `\extrasfrench` and `\noextrasfrench`; we will just redefine `\shorthandoff` and `\shorthandon` in French to issue a warning reminding the user that active characters are no longer used in French with recent LuaTeX engines.

```
\ifFB@luatex@punct
  \newcommand*{\FB@luatex@punct@french}{%
     \babel@save{\shorthandon}%
     \babel@save{\shorthandoff}%
     \def\shorthandoff##1{%
        \ifx\PackageWarning\@undefined
          \fb@warning{\noexpand\shorthandoff{;:!?} is helpless with
           LuaTeX,\\ use \noexpand\NoAutoSpacing
           *inside a group* instead.}%
        \else
          \PackageWarning{frenchb.ldf}{\protect\shorthandoff{;:!?} is
           helpless with LuaTeX,\MessageBreak use \protect\NoAutoSpacing
           \space *inside a group* instead;\MessageBreak reported}%
        \fi}%
     \def\shorthandon##1{}%
  }
  \FB@addto{extras}{\FB@luatex@punct@french}
```

In $\LaTeX 2_\varepsilon$, file `frenchb.lua` will be loaded 'AtBeginDocument' *after* processing options (`ThinColonSpace` needs to be taken into account). The next definition will be used to activate Lua punctuation: it sets the language number for French, loads `frenchb.lua` and adds function `french_punctuation` at the end of the `kerning` callback (no priority).

```
  \def\activate@luatexpunct{%
    \directlua{%
      FR = \the\l@french
      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 2.10).
The default value for `\XeTeXcharclass` is 0 for characters tokens and `\FB@nonchar` for all other tokens (glues, kerns, math and box boundaries, etc.). These defaults should not be changed otherwise the spacing before the 'high punctuation' characters and inside quotes might not be correct.
We switch `\XeTeXinterchartokenstate` to 1 and change the `\XeTeXcharclass` values of `;` `!` `?` `:` `(` `]` « and » when entering French. Special care is taken to restore them to their inital values when leaving French.
The following part holds specific code for punctuation with XeTeX engines.

```
\ifFB@xetex@punct
   \ifLaTeXe
    \PackageInfo{frenchb.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@charlist` holds the all list of characters which have their `\XeTeXcharclass` value modified in French: the first set includes high punctuation, French quotes, opening delimiters and no-break spaces

"21	"3A	"3B	"3F	"AB	"BB	"28	"5B	"A0	"202F
!	:	;	?	«	»	(	[		

the second one holds those which need resetting in French when `xeCJK.sty` is in use

"29	"5D	"7B	"7D	"2C	"2D	"2E	"22	"25	"27	"60	"2019
)	]	{	}	,	-	.	"	%	'	‘	’

```
\def\FB@charlist{"21,"3A,"3B,"3F,"AB,"BB,"28,"5B,"A0,"202F,%
                 "29,"5D,"7B,"7D,"2C,"2D,"2E,"22,"25,"27,"60,"2019}
```

`\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. It also redefines `\shorthandoff` and `\shorthandon` (locally) to avoid error messages with XeTeX-based engines.

```
\newcommand*{\FB@xetex@punct@french}{%
  \babel@savevariable{\XeTeXinterchartokenstate}%
  \babel@save{\shorthandon}%
  \babel@save{\shorthandoff}%
  \bbl@for\FB@char\FB@charlist
       {\FBsavevariable@loop{\XeTeXcharclass}{\FB@char}}%
  \def\shorthandoff##1{%
    \ifx\PackageWarning\@undefined
      \fb@warning{\noexpand\shorthandoff{;:!?} is helpless with
        XeTeX,\\ use \noexpand\NoAutoSpacing
        *inside a group* instead.}%
    \else
      \PackageWarning{frenchb.ldf}{\protect\shorthandoff{;:!?} is
        helpless with XeTeX,\MessageBreak use \protect\NoAutoSpacing
        \space *inside a group* instead;\MessageBreak reported}%
    \fi}%
   \def\shorthandon##1{}%
```

Let’s now set the classes 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=1
  \XeTeXcharclass `\: = \FB@punctthick
  \XeTeXinterchartoks \z@ \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: unfortunately `\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 \z@ \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 \z@ = {%
        \ifFB@spacing\FB@guillspace\fi}%
  \XeTeXinterchartoks \FB@guilo \FB@nonchar = {%
        \ifFB@spacing\FB@guillspace\ignorespaces\fi}%
```

```
     \XeTeXinterchartoks \z@ \FB@guilf = {%
           \ifFB@spacing\FB@guillspace\fi}%
     \XeTeXinterchartoks \FB@puncthin \FB@guilf = {%
           \ifFB@spacing\FB@guillspace\fi}%
     \XeTeXinterchartoks \FB@nonchar \FB@guilf = {%
           \ifFB@spacing\unskip\FB@guillspace\fi}%
```

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

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

These characters have their class changed by `xeCJK.sty`, let's reset them to 0 in French.

```
     \bbl@for\FB@char
             {`\{,`\,,`\.,`\-,`\),`\],`\},`\%,"22,"27,"60,"2019}%
             {\XeTeXcharclass\FB@char=\z@}%
   }
   \FB@addto{extras}{\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.

```
\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 an unbreakable `\FBthinspace` instead. If no space has been typed, we add `\FDP@thinspace` which will be defined, up to the user's wishes, as `\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.

```
\FB@addto{extras}{\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{?}%
  }
  \FB@addto{noextras}{%
    \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 unbreakable 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=1 \fi
         \def\FDP@thinspace{\penalty\@M\FBthinspace}%
         \def\FDP@colonspace{\penalty\@M\FBcolonspace}}
\def\noautospace@beforeFDP{%
         \ifFB@luatex@punct\FB@addDPspace=0 \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 LaTeX 2$_\varepsilon$ `\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

With pdfLaTeX LaTeX 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
```

\FB@og
\FB@fg
\FBguillspace

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. \FBguillspace which does the spacing, has been fine tuned by Thierry Bouche to 80% of an inter-word space with reduced stretchability. French quotes (including spacing) are printed by \FB@og and \FB@fg, the expansion of the top level commands \og and \og is different in and outside French.
LuaTeX requires toks; \FBguillsp will be computed from \FBguillspace 'AtBegin-Document', its dimensions will be scaled by frenchb.lua for the current font and used after '«' and before '»'.

```
\newcommand*{\FBguillspace}{\hskip.8\fontdimen2\font
                              plus.3\fontdimen3\font
                             minus.8\fontdimen4\font \relax}
\newcommand*{\FB@guillspace}{\penalty\@M\FBguillspace}
\newtoks\FBguillsp
```

The definitions of \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.

```
\ifFB@luatex@punct
  \DeclareRobustCommand*{\FB@og}{\leavevmode
          \bgroup\FB@spacing=0 \guillemotleft\egroup
          \FB@guillspace}
  \DeclareRobustCommand*{\FB@fg}{\ifdim\lastskip>\z@\unskip\fi
          \FB@guillspace
          \bgroup\FB@spacing=0 \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

\FB@addto{extras}{\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*{\ogii}{\textquotedblleft}
\newcommand*{\fgii}{\textquotedblright}
```

and the needed technical stuff to handle options:

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

The main command `\frquote` accepts (in LaTeX 2_ε 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.

```
\ifLaTeXe
```

```
  \DeclareRobustCommand\frquote{%
       \@ifstar{\FBcloseguillfalse\fr@quote}%
               {\FBcloseguilltrue\fr@quote}}
\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: save `\everypar` before customising it, set `\FBeverypar@quote` for level 1 quotations and add it to `\everypar`, then print the quotation:

```
    \FB@everypar=\everypar
    \ifx\FBeveryparguill\FBguillnone
    \else
      \def\FBeverypar@quote{\FBeveryparguill\FB@guillspace}%
      \everypar=\expandafter{\the\everypar \FBeverypar@quote}%
    \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
      \localleftbox{\guillemotleft\FB@guillspace}%
      \let\FBeverypar@quote\relax
      \ogi #1\ifFBcloseguill\fgi\fi
    \else
      \ifx\FBeverylineguill\FBguillclose
        \localleftbox{\guillemotright\FB@guillspace}%
        \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 ≥ 3:

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

Clean on exit: adjust \FBguill@level and restore \localleftbox and \everypar.

```
  \advance\FBguill@level by \m@ne
  \ifx\FBeverylineguill\FBguillnone\else\localleftbox{}\fi
  \ifx\FBeveryparguill\FBguillnone\else\everypar=\FB@everypar\fi
}
```

2.4 Date in French

\datefrench The macro \datefrench redefines the command \today to produce French dates. This new implementation requires babel 3.9i or newer but, as of 3.9k, doesn't work with Plain based formats, so \date\CurrentOption is defined the old way for these formats.

```
\ifLaTeXe
  \def\BabelLanguages{french,acadian}
  \StartBabelCommands*{\BabelLanguages}{date}
      [unicode, fontenc=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{{\number\day}\ifnum1=\day {\ier}\fi\space
        \csname month\romannumeral\month name\endcsname \space
        \number\year
       }
  \EndBabelCommands
\else
  \ifFBunicode
    \@namedef{date\CurrentOption}{%
      \def\today{{\number\day}\ifnum1=\day {\ier}\fi \space
          \ifcase\month
      \or janvier\or février\or mars\or avril\or mai\or
      juin\or juillet\or août\or septembre\or
      octobre\or novembre\or décembre\fi
     \space \number\year}}
 \else
  \@namedef{date\CurrentOption}{%
   \def\today{{\number\day}\ifnum1=\day {\ier}\fi \space
    \ifcase\month
     \or janvier\or f\'evrier\or mars\or avril\or mai\or
     juin\or juillet\or ao\^ut\or septembre\or
     octobre\or novembre\or d\'ecembre\fi
    \space \number\year}}
 \fi
\fi
```

2.5 Extra utilities

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

\up \fup

\up eases the typesetting of superscripts like ‘1$^{\text{er}}$’. Up to version 2.0 of babel-french \up was just a shortcut for \textsuperscript in LaTeX 2$_{\varepsilon}$, 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`.

```
\DeclareRobustCommand*{\fup}[1]{%
  \ifx\realsuperscript\@undefined
    \FB@up{#1}%
  \else
    \bgroup\let\fakesuperscript\FB@up@fake
```

```
              \realsuperscript{\FB@lc{#1}}\egroup
    \fi}
```

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

```
  \providecommand*{\up}{\relax}
```

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}
```

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

And some more macros relying on `\up` for numbering, first two support macros.

```
\newcommand*{\FrenchEnumerate}[1]{%
                           #1\up{o}\kern+.3em}
\newcommand*{\FrenchPopularEnumerate}[1]{%
                           #1\up{o})\kern+.3em}
```

Typing `\primo` should result in ‘^{o} ’,

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

while typing `\fprimo)` gives ‘^{o}) .

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

Let’s provide four macros for the common abbreviations of “Numéro”.

```
\DeclareRobustCommand*{\No}{N\up{o}\kern+.2em}
\DeclareRobustCommand*{\no}{n\up{o}\kern+.2em}
\DeclareRobustCommand*{\Nos}{N\up{os}\kern+.2em}
\DeclareRobustCommand*{\nos}{n\up{os}\kern+.2em}
```

`\bsc`

As family names should be written in small capitals and never be hyphenated, we provide a command (its name comes from Boxed Small Caps) to input them easily. Note that this command has changed with version 2 of babel-french: a `\kern0pt` is used instead of `\hbox` because `\hbox` would break microtype’s font expansion; as a (positive?) side effect, composed names (such as Dupont-Durand) can now be hyphenated on explicit hyphens. Usage: `Jean~\bsc{Duchemin}`.

```
\DeclareRobustCommand*{\bsc}[1]{\leavevmode\begingroup\kern0pt
                                          \scshape #1\endgroup}
\ifLaTeXe\else\let\scshape\relax\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
  \newcommand*{\at}{{\char"0040}}
  \newcommand*{\circonflexe}{{\char"005E}}
  \newcommand*{\tild}{{\char"007E}}
  \newcommand*{\boi}{{\char"005C}}
  \newcommand*{\degre}{{\char"00B0}}
\else
  \ifLaTeXe
    \DeclareTextSymbol{\at}{T1}{64}
    \DeclareTextSymbol{\circonflexe}{T1}{94}
    \DeclareTextSymbol{\tild}{T1}{126}
    \DeclareTextSymbolDefault{\at}{T1}
    \DeclareTextSymbolDefault{\circonflexe}{T1}
    \DeclareTextSymbolDefault{\tild}{T1}
    \DeclareRobustCommand*{\boi}{\textbackslash}
    \DeclareRobustCommand*{\degre}{\r{}}
  \else
    \def\T@one{T1}
    \ifx\f@encoding\T@one
      \newcommand*{\degre}{{\char6}}
    \else
      \newcommand*{\degre}{{\char23}}
    \fi
    \newcommand*{\at}{{\char64}}
    \newcommand*{\circonflexe}{{\char94}}
    \newcommand*{\tild}{{\char126}}
    \newcommand*{\boi}{$\backslash$}
  \fi
\fi
```

`\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`).
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.

```
\ifLaTeXe
  \newcommand*{\degres}{\degre}
  \ifFBunicode
    \DeclareRobustCommand*{\degres}{\degre}
  \else
    \def\Warning@degree@TSone{\FBWarning
              {Degrees would look better in TS1-encoding:%
               \MessageBreak add \protect
              \usepackage{textcomp} to the preamble.%
              \MessageBreak Degrees used}}
    \AtBeginDocument{\ifx\DeclareEncodingSubset\@undefined
                       \DeclareRobustCommand*{\degres}{%
                          \leavevmode\hbox to 0.3em{\hss\degre\hss}%
                       \Warning@degree@TSone
                       \global\let\Warning@degree@TSone\relax}%
                     \else
                       \DeclareRobustCommand*{\degres}{%
                          \hbox{\UseTextSymbol{TS1}{\textdegree}}}%
                     \fi
                     }
  \fi
\else
  \newcommand*{\degres}{%
    \leavevmode\hbox to 0.3em{\hss\degre\hss}}
\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 *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
\newcommand*{\DecimalMathComma}{%
  \iflanguage{french}{\dec@math@comma}{}%
  \ifFB@icomma\else\FB@addto{extras}{\dec@math@comma}\fi
}
\newcommand*{\StandardMathComma}{%
  \std@math@comma
  \ifFB@icomma\else\FB@addto{extras}{\std@math@comma}\fi
}
\ifLaTeXe
  \AtBeginDocument{\@ifpackageloaded{icomma}%
                    {\FB@icommatrue}%
                    {\FB@addto{noextras}{\std@math@comma}}%
  }
\else
  \FB@addto{noextras}{\std@math@comma}
\fi
```

`\nombre` The command `\nombre` is now borrowed from `numprint.sty` for LaTeX 2ε. There is no point to maintain the former tricky code when a package is dedicated to do the same job and more. For Plain based formats, `\nombre` no longer formats numbers, it prints them as is and issues a warning about the change.
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! ***}}
```

The next definitions only make sense for LaTeX 2ε. For Plain based formats, let's activate LuaTeX punctuation if necessary, then cleanup and exit. Temporary fix: `\l@french` is not properly set by babel 3.9h with Plain LuaTeX format.

```
\let\FBstop@here\relax
\def\FBclean@on@exit{\let\ifLaTeXe\undefined
                     \let\LaTeXetrue\undefined
                     \let\LaTeXefalse\undefined}
\ifx\magnification\@undefined
\else
  \def\FBstop@here{\ifFB@luatex@punct
                     \activate@luatexpunct
                   \fi
                   \FBclean@on@exit
                   \ldf@quit\CurrentOption\endinput}
\fi
\FBstop@here
```

What follows is for LaTeX 2ε *only*; as all LaTeX 2ε based formats include ε-TeX, we can use `\ifdefined` now. We redefine `\nombre` for LaTeX 2ε. 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{frenchb.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
}
```

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.
Let's give a chance to a class or a package read before frenchb to define \FBfigtabshape as \relax, otherwise \FBfigtabshape will be defined as \scshape (can be changed with \frenchsetup{SmallCapsFigTabCaptions=false}).

```
\ifx\FBfigtabshape\@undefined \let\FBfigtabshape\scshape \fi
```

New implementation for caption names (requires babel's 3.9 or up).

```
\StartBabelCommands*{\BabelLanguages}{captions}
      [unicode, fontenc=EU1 EU2 TU, 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}{{\FBfigtabshape Figure}}
  \SetString{\tablename}{{\FBfigtabshape 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}{%
    \frenchpartfirst,\frenchpartsecond,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{}}%
    \DeclareRobustCommand*{\FB@partname}{%
       \ifFBPartNameFull
         \csname ordinal\romannumeral\value{part}\endcsname\space
         \frenchpartnameord\FB@emptypart
       \else
         Partie%
       \fi}%
   }
  \SetString{\partname}{\FB@partname}
\EndBabelCommands
```

The following patch is for koma-script classes: \partformat needs to be redefined in French as this command, defined as \partname~\thepart\autodot is incompatible with our redefinition of \partname. The code is postponed to the end of package because \ifFB@koma will be defined and set later on (see p. 44).

```
\AtEndOfPackage{%
  \ifFB@koma
    \ifdefined\partformat
      \FB@addto{captions}{%
```

```
          \ifFBPartNameFull
            \babel@save\partformat
            \renewcommand*{\partformat}{\partname}%
          \fi}%
      \fi
    \fi
}
```

Up to v2.6h babel-french used to merge `\captionsfrenchb` and `\captionsfrancais` into `\captionsfrench` at `\begin{document}`. This is deprecated in favor of the new (much simpler!) syntax introduced in babel 3.9. No need to define `\captionscanadien` and `\captionsacadian` either.

`\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 LaTeX 2_ε classes; the ':' is made active too late, no space is added before it. 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 LaTeX 2_ε 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.

```
\newif\if@FBwarning@capsep
\@FBwarning@capseptrue
\newcommand{\FBWarning}[1]{\PackageWarning{frenchb.ldf}{#1}}
\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 unless high punctuation is active.

```
\ifFB@active@punct\else\@FBwarning@capsepfalse\fi
```

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`) .

```
\newif\ifFB@koma
\@ifclassloaded{memoir}{\@FBwarning@capsepfalse}{}
\@ifclassloaded{scrartcl}{\@FBwarning@capsepfalse\FB@komatrue}{}
\@ifclassloaded{scrbook}{\@FBwarning@capsepfalse\FB@komatrue}{}
\@ifclassloaded{scrreprt}{\@FBwarning@capsepfalse\FB@komatrue}{}
```

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
```

The `caption`, `subcaption` and `floatrow` packages are compatible with babel-french if they are loaded after babel.
Check if package `caption` is loaded now (before babel-french), then issue a warning advising to load it after babel-french and disable the standard warning.

```
\@ifpackageloaded{caption}
   {\FBWarning{Please load the "caption" package\MessageBreak
               AFTER babel/frenchb; reported}%
    \@FBwarning@capsepfalse}%
   {}
```

Same for package subcaption.

```
\@ifpackageloaded{subcaption}
   {\FBWarning{Please load the "subcaption" package\MessageBreak
               AFTER babel/frenchb; reported}%
    \@FBwarning@capsepfalse}%
   {}
```

Same for package floatrow.

```
\@ifpackageloaded{floatrow}
   {\FBWarning{Please load the "floatrow" package\MessageBreak
               AFTER babel/frenchb; reported}%
    \@FBwarning@capsepfalse}%
   {}
```

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
```

Do not overwrite \FBCaption@Separator if already saved as ‘: ’ for other languages and set to \CaptionSeparator by \extrasfrench when French is the main language.

```
    \ifFBOldFigTabCaptions
    \else
      \def\FBCaption@Separator{{\autospace@beforeFDP : }}%
    \fi
    \ifFBCustomiseFigTabCaptions
      \ifx\bbl@main@language\FB@french
        \def\FBCaption@Separator{\CaptionSeparator}%
      \fi
    \fi
    \@FBwarning@capsepfalse
  \fi
  \if@FBwarning@capsep
    \FBWarning
       {Figures’ and tables’ captions might look like\MessageBreak
        ‘Figure 1:’ which is wrong in French.\MessageBreak
        Check your class or packages to change this;\MessageBreak
        reported}%
  \fi
  \let\FB@makecaption\relax
  \let\STD@makecaption\relax
}
```

2.8 Dots...

\FBtextellipsis LaTeX 2ε’s standard definition of \dots in text-mode is \textellipsis which 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 2ε only).
The `\if` construction in the LaTeX 2ε definition of `\dots` doesn't allow the use of `xspace` (`xspace` is always followed by a `\fi`), so we use the AMS-LaTeX construction of `\dots`; this has to be done 'AtBeginDocument' not to be overwritten when `amsmath.sty` is loaded after babel.
LY1 has a ready made character for `\textellipsis`, it should be used in French too. The same is true for Unicode fonts in use with XeTeX and LuaTeX.

```
\ifFBunicode
  \let\FBtextellipsis\textellipsis
\else
  \DeclareTextSymbol{\FBtextellipsis}{LY1}{133}
  \DeclareTextCommandDefault{\FBtextellipsis}{%
    .\kern\fontdimen3\font.\kern\fontdimen3\font.\xspace}
\fi
```

`\Mdots@` and `\Tdots@` hold the definitions of `\dots` in Math and Text mode. They default to those of amsmath-2.0, and will revert to standard LaTeX definitions 'AtBeginDocument', if amsmath has not been loaded. `\Mdots@` doesn't change when switching from/to French, while `\Tdots@` is redefined as `\FBtextellipsis` in French.

```
\newcommand*{\Tdots@}{\@xp\textellipsis}
\newcommand*{\Mdots@}{\@xp\mdots@}
\AtBeginDocument{\DeclareRobustCommand*{\dots}{\relax
                 \csname\ifmmode M\else T\fi dots@\endcsname}%
                 \ifdefined\@xp\else\let\@xp\relax\fi
                 \ifdefined\mdots@\else\let\Mdots@\mathellipsis\fi
                }
\def\bbl@frenchdots{\babel@save\Tdots@ \let\Tdots@\FBtextellipsis}
\FB@addto{extras}{\bbl@frenchdots}
```

2.9 More checks about packages' loading order

Like packages captions and `floatrow` (see section 2.7), package `listings` should be loaded after babel-french due to active characters issues (pdfLaTeX only).

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

Package `beamerarticle` should be loaded before babel-french to avoid list's conflicts, see p. 48.

```
\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/frenchb; reported}%
  \fi
}
```

2.10 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 'AtEndOfPackage' 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                \FBShowOptionsfalse
\newif\ifFBStandardLayout             \FBStandardLayouttrue
\newif\ifFBGlobalLayoutFrench         \FBGlobalLayoutFrenchtrue
\newif\ifFBReduceListSpacing          \FBReduceListSpacingfalse
\newif\ifFBListOldLayout              \FBListOldLayoutfalse
\newif\ifFBCompactItemize             \FBCompactItemizefalse
\newif\ifFBStandardItemizeEnv         \FBStandardItemizeEnvtrue
\newif\ifFBStandardEnumerateEnv       \FBStandardEnumerateEnvtrue
\newif\ifFBStandardItemLabels         \FBStandardItemLabelstrue
\newif\ifFBStandardLists              \FBStandardListstrue
\newif\ifFBIndentFirst                \FBIndentFirstfalse
\newif\ifFBFrenchFootnotes            \FBFrenchFootnotesfalse
\newif\ifFBAutoSpaceFootnotes         \FBAutoSpaceFootnotesfalse
\newif\ifFBOriginalTypewriter         \FBOriginalTypewriterfalse
\newif\ifFBThinColonSpace             \FBThinColonSpacefalse
\newif\ifFBThinSpaceInFrenchNumbers \FBThinSpaceInFrenchNumbersfalse
\newif\ifFBFrenchSuperscripts         \FBFrenchSuperscriptstrue
\newif\ifFBLowercaseSuperscripts      \FBLowercaseSuperscriptstrue
\newif\ifFBPartNameFull               \FBPartNameFulltrue
\newif\ifFBCustomiseFigTabCaptions  \FBCustomiseFigTabCaptionsfalse
\newif\ifFBOldFigTabCaptions          \FBOldFigTabCaptionsfalse
\newif\ifFBSmallCapsFigTabCaptions  \FBSmallCapsFigTabCaptionstrue
\newif\ifFBSuppressWarning            \FBSuppressWarningfalse
\newif\ifFBINGuillSpace               \FBINGuillSpacefalse
```

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, 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{}`.
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 2.9; a light customisation is compatible with the `beamerarticle` package.

```
\edef\FB@french{\CurrentOption}
\AtEndOfPackage{%
  \ifx\bbl@main@language\FB@french
    \FBGlobalLayoutFrenchtrue
    \@ifclassloaded{beamer}%
      {\PackageInfo{frenchb.ldf}{%
          No list customisation for the beamer class,%
          \MessageBreak reported}}%
      {\@ifpackageloaded{beamerarticle}%
         {\FBStandardItemLabelsfalse
          \FBReduceListSpacingtrue
```

```
        \PackageInfo{frenchb.ldf}{%
           Minimal list customisation for the beamerarticle%
           \MessageBreak package; reported}}%
```

Otherwise customise lists “à la française”:

```
       {\FBReduceListSpacingtrue
        \FBStandardItemizeEnvfalse
        \FBStandardEnumerateEnvfalse
        \FBStandardItemLabelsfalse}%
    }
  \FBIndentFirsttrue
  \FBFrenchFootnotestrue
  \FBAutoSpaceFootnotestrue
  \FBCustomiseFigTabCaptionstrue
\else
  \FBGlobalLayoutFrenchfalse
\fi
```

babel-french being an option of babel, it cannot load a package (`keyval`) while `frenchb.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}%
\define@key{FB}{StandardLayout}[true]%
        {\csname FBStandardLayout#1\endcsname
         \ifFBStandardLayout
           \FBReduceListSpacingfalse
           \FBStandardItemizeEnvtrue
           \FBStandardItemLabelstrue
           \FBStandardEnumerateEnvtrue
           \FBIndentFirstfalse
           \FBFrenchFootnotesfalse
           \FBAutoSpaceFootnotesfalse
           \FBGlobalLayoutFrenchfalse
         \else
           \FBReduceListSpacingtrue
           \FBStandardItemizeEnvfalse
           \FBStandardItemLabelsfalse
           \FBStandardEnumerateEnvfalse
           \FBIndentFirsttrue
           \FBFrenchFootnotestrue
           \FBAutoSpaceFootnotestrue
         \fi}%
\define@key{FB}{GlobalLayoutFrench}[true]%
        {\csname FBGlobalLayoutFrench#1\endcsname
```

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. Warn and reset in case this key is set to true while the main language is *not* French.

```
         \ifFBGlobalLayoutFrench
```

```
           \ifx\bbl@main@language\FB@french
           \else
             \FBGlobalLayoutFrenchfalse
             \PackageWarning{frenchb.ldf}%
                {Option ‘GlobalLayoutFrench’ skipped:\MessageBreak
                 French is *not* babel’s last option.\MessageBreak
                 Reported}%
           \fi
         \fi}%
\define@key{FB}{ReduceListSpacing}[true]%
        {\csname FBReduceListSpacing#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
           \FBReduceListSpacingfalse
           \FBCompactItemizefalse
           \FBStandardItemizeEnvtrue
           \FBStandardEnumerateEnvtrue
           \FBStandardItemLabelstrue
         \else
```

```
          \FBReduceListSpacingtrue
          \FBCompactItemizetrue
          \FBStandardItemizeEnvfalse
          \FBStandardEnumerateEnvfalse
          \FBStandardItemLabelsfalse
        \fi}%
\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
```

`\CurrentOption` no longer defined. It's value has been saved in `\FB@CurOpt` while reading `frenchb.ldf`.

```
         \ifFBOldFigTabCaptions
           \FB@addto{extras}{\babel@save\FBCaption@Separator
                    \def\FBCaption@Separator{\CaptionSeparator}}%
         \fi}%
\define@key{FB}{SmallCapsFigTabCaptions}[true]%
        {\csname FBSmallCapsFigTabCaptions#1\endcsname
         \ifFBSmallCapsFigTabCaptions
           \let\FBfigtabshape\scshape
         \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
                     \PackageWarning{frenchb.ldf}%
                        {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}%
```

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`. 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 unbreakable 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 `inputenc` package has to be loaded before the `\begin{document}` with the proper coding option, so we check if `\DeclareInputText` is defined.
Life is much simpler 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.

```
\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
             \PackageWarning{frenchb.ldf}%
                {Option og=« not supported with this version
                 of\MessageBreak LuaTeX/XeTeX; reported}%
           \fi
         \else
```

This is for conventional TeX engines:

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

Package inputenc with utf8x encoding loaded, use `\uc@dclc`,

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

if encoding is not utf8x, try utf8...

```
                  \ifdefined\DeclareUnicodeCharacter
```

utf8 loaded, use `\DeclareUnicodeCharacter`,

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

if utf8 is not loaded either, we assume 8-bit character input encoding. Package MULEenc (from CJK) defines `\mule@def` to map characters to control sequences.

```
                    \@tempcnta‘#1\relax
                    \ifdefined\mule@def
                      \mule@def{11}{\FB@@og}%
                    \else
                      \DeclareInputText{\the\@tempcnta}{\FB@@og}%
                    \fi
                  \fi
                \fi
              \else
```

Package inputenc not loaded, no way...

```
                \PackageWarning{frenchb.ldf}%
                   {Option ‘og’ requires package inputenc;%
                    \MessageBreak reported}%
              \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
             \PackageWarning{frenchb.ldf}%
                {Option fg=» not supported with this version
                 of\MessageBreak LuaTeX/XeTeX; reported}%
           \fi
         \else
           \newcommand*{\FB@@fg}{%
              \iflanguage{french}%
                {\ifFB@spacing\FB@fg
                 \else\guillemotright
                 \fi}%
                {\guillemotright}}%
```

```
            \AtBeginDocument{%
              \ifdefined\DeclareInputText
                \ifdefined\uc@dclc
                  \uc@dclc{187}{default}{\FB@@fg}%
                \else
                  \ifdefined\DeclareUnicodeCharacter
                    \DeclareUnicodeCharacter{00BB}{\FB@@fg}%
                  \else
                    \@tempcnta`#1\relax
                    \ifdefined\mule@def
                      \mule@def{27}{{\FB@@fg}}%
                    \else
                      \DeclareInputText{\the\@tempcnta}{\FB@@fg}%
                    \fi
                  \fi
                \fi
              \else
                \PackageWarning{frenchb.ldf}%
                   {Option `fg' requires package inputenc;%
                    \MessageBreak reported}%
              \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{frenchb.ldf}%
          {Setting StandardItemizeEnv=true for\MessageBreak
           compatibility with enumitem package,\MessageBreak
           reported}%
     \fi
     \ifFBStandardEnumerateEnv
     \else
       \FBStandardEnumerateEnvtrue
       \PackageInfo{frenchb.ldf}%
          {Setting StandardEnumerateEnv=true for\MessageBreak
           compatibility with enumitem package,\MessageBreak
           reported}%
     \fi}{}%
  \@ifpackageloaded{paralist}{%
     \ifFBStandardItemizeEnv
     \else
       \FBStandardItemizeEnvtrue
       \PackageInfo{frenchb.ldf}%
          {Setting StandardItemizeEnv=true for\MessageBreak
           compatibility with paralist package,\MessageBreak
           reported}%
     \fi
     \ifFBStandardEnumerateEnv
     \else
       \FBStandardEnumerateEnvtrue
       \PackageInfo{frenchb.ldf}%
          {Setting StandardEnumerateEnv=true for\MessageBreak
           compatibility with paralist package,\MessageBreak
           reported}%
     \fi}{}%
  \@ifpackageloaded{enumerate}{%
     \ifFBStandardEnumerateEnv
     \else
       \FBStandardEnumerateEnvtrue
       \PackageInfo{frenchb.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}
  \ifx\bbl@main@language\FB@french
    \update@frenchlists
  \fi
```

The layout of footnotes is handled at the `\begin{document}` depending on the values of flags `FrenchFootnotes` and `AutoSpaceFootnotes` (see section 2.13), nothing has to be done here for footnotes.
`AutoSpacePunctuation` adds an unbreakable 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 have to provide this command.

```
\@ifpackageloaded{numprint}%
{\ifnprt@autolanguage
   \providecommand*{\npstylefrench}{}%
   \ifFBThinSpaceInFrenchNumbers
     \renewcommand*\npstylefrench{%
        \npthousandsep{\,}%
        \npdecimalsign{,}%
        \npproductsign{\cdot}%
        \npunitseparator{\,}%
        \npdegreeseparator{}%
        \nppercentseparator{\nprt@unitsep}%
        }%
   \else
     \renewcommand*\npstylefrench{%
        \npthousandsep{~}%
        \npdecimalsign{,}%
        \npproductsign{\cdot}%
        \npunitseparator{\,}%
        \npdegreeseparator{}%
        \nppercentseparator{\nprt@unitsep}%
        }%
   \fi
   \npaddtolanguage{french}{french}%
 \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}{\@ifstar{\FB@up@fake}{\fup}}%
\else
  \DeclareRobustCommand*{\up}{\@ifstar{\FB@up@fake}%
                                        {\textsuperscript}}%
\fi
```

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

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

```
\fi
```

Unless CustomiseFigTabCaptions has been set to false, use \CaptionSeparator for koma-script, memoir and beamer classes.

```
\ifFBCustomiseFigTabCaptions
  \ifFB@koma
    \renewcommand*{\captionformat}{\CaptionSeparator}%
  \fi
  \@ifclassloaded{memoir}%
     {\captiondelim{\CaptionSeparator}}{}%
  \@ifclassloaded{beamer}%
     {\defbeamertemplate{caption label separator}{FBcustom}{%
          \CaptionSeparator}%
      \setbeamertemplate{caption label separator}[FBcustom]}{}%
\else
```

When CustomiseFigTabCaptions is false, have the colon behave properly in French: locally force \autospace@beforeFDP in case of AutoSpacePunctuation=false.

```
  \ifFB@koma
    \renewcommand*{\captionformat}{{\autospace@beforeFDP : }}%
  \fi
  \@ifclassloaded{memoir}%
     {\captiondelim{{\autospace@beforeFDP : }}%
     }{}%
  \@ifclassloaded{beamer}%
     {\defbeamertemplate{caption label separator}{FBcolon}{%
           {\autospace@beforeFDP : }}%
      \setbeamertemplate{caption label separator}[FBcolon]%
     }{}%
\fi
```

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

```
\ifFBShowOptions
  \GenericWarning{* }{%
   * **** List of possible options for frenchb ****\MessageBreak
   [Default values between brackets when frenchb is loaded *LAST*]%
   \MessageBreak
   ShowOptions=true [false]\MessageBreak
   StandardLayout=true [false]\MessageBreak
   GlobalLayoutFrench=false [true]\MessageBreak
   StandardLists=true [false]\MessageBreak
   IndentFirst=false [true]\MessageBreak
   ReduceListSpacing=false [true]\MessageBreak
   ListOldLayout=true [false]\MessageBreak
   StandardItemizeEnv=true [false]\MessageBreak
   StandardEnumerateEnv=true [false]\MessageBreak
   StandardItemLabels=true [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
   FrenchFootnotes=false [true]\MessageBreak
   AutoSpaceFootnotes=false [true]\MessageBreak
   AutoSpacePunctuation=false [true]\MessageBreak
   OriginalTypewriter=true [false]\MessageBreak
   ThinColonSpace=true [false]\MessageBreak
   ThinSpaceInFrenchNumbers=true [false]\MessageBreak
   FrenchSuperscripts=false [true]\MessageBreak
   LowercaseSuperscripts=false [true]\MessageBreak
   PartNameFull=false [true]\MessageBreak
   SuppressWarning=true [false]\MessageBreak
   CustomiseFigTabCaptions=false [true]\MessageBreak
   OldFigTabCaptions=true [false]\MessageBreak
   SmallCapsFigTabCaptions=false [true]\MessageBreak
   INGuillSpace=true [false]\MessageBreak
   InnerGuillSingle=true [false]\MessageBreak
   EveryParGuill=open, close, none [open]\MessageBreak
   EveryLineGuill=open, close, none
                [open in LuaTeX, none otherwise]\MessageBreak
   og= <left quote character>, fg= <right quote character>%
   \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 redefine some commands in hyperref's bookmarks.

```
  \ifdefined\pdfstringdefDisableCommands
    \pdfstringdefDisableCommands{%
      \let\up\relax
      \let\fup\relax
      \let\degre\textdegree
      \let\degres\textdegree
      \def\ieme{e\xspace}%
      \def\iemes{es\xspace}%
      \def\ier{er\xspace}%
      \def\iers{ers\xspace}%
      \def\iere{re\xspace}%
      \def\ieres{res\xspace}%
      \def\FrenchEnumerate#1{#1\degre\space}%
      \def\FrenchPopularEnumerate#1{#1\degre)\space}%
      \def\No{N\degre\space}%
      \def\no{n\degre\space}%
      \def\Nos{N\degre\space}%
      \def\nos{n\degre\space}%
      \def\FB@og{\guillemotleft\space}%
      \def\FB@fg{\space\guillemotright}%
      \def\at{@}%
      \def\circonflexe{\string^}%
      \def\tild{\string~}%
      \def\boi{\textbackslash}%
      \let\bsc\textsc
    }%
  \fi
```

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

```
  \FBprocess@options
```

The final definitions of commands ruling spacing in French been known, let’s reset the corresponding toks for LuaTeX and load file `frenchb.lua` (LuaTeX only).

```
  \ifFB@luatex@punct
    \FBcolonsp=\expandafter{\meaning\FBcolonspace}
    \FBthinsp= \expandafter{\meaning\FBthinspace}
    \FBguillsp=\expandafter{\meaning\FBguillspace}
    \activate@luatexpunct
  \fi
```

Some warnings are issued when output font encodings are not properly set. With XeLaTeX or LuaLaTeX, `fontspec.sty` should be loaded unless T1 encoded fonts are used through `luainputenc`, in the latter case `\FB@og` and `\FB@fg` have to be redefined; with (pdf)LaTeX, a warning is issued when OT1 encoding is in use at the `\begin{document}`. Mind that `\encodingdefault` is defined as ‘long’, defining `\FBOTone` with `\newcommand*` would fail!

```
  \ifFBunicode
    \@ifpackageloaded{fontspec}{}%
      {\@ifpackageloaded{luainputenc}{}%
          {\PackageWarning{frenchb.ldf}%
              {Add \protect\usepackage{fontspec} to the\MessageBreak
               preamble of your document, reported}%
          }%
      }
  \else
    \begingroup \newcommand{\FBOTone}{OT1}%
    \ifx\encodingdefault\FBOTone
      \PackageWarning{frenchb.ldf}%
         {OT1 encoding should not be used for French.%
          \MessageBreak
          Add \protect\usepackage[T1]{fontenc} to the
          preamble\MessageBreak of your document; reported}%
    \fi
    \endgroup
  \fi
}
```

2.11 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
\def\FB@listVsettings{%
      \setlength{\itemsep}{0.4ex plus 0.2ex minus 0.2ex}%
      \setlength{\parsep}{0.4ex plus 0.2ex minus 0.2ex}%
      \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 `\@tempdima`.

```
      \@tempdima=\parskip
      \addtolength{\topsep}{-\@tempdima}%
      \addtolength{\partopsep}{\@tempdima}%
}
\def\listFB#1#2{\listORI{#1}{\FB@listVsettings #2}}
\let\endlistFB\endlist
```

Let's now consider French itemize-lists. They differ from those provided by the standard LaTeX 2_ε 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 is stored in `\FrenchLabelItem}`, it defaults to '—' and can be changed using `\frenchsetup{}` (see section 2.10).
- 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 follows:

```
   Text starting at ‘parindent’
⇐ Leftmargin
   — first item. . .
      — first second level item
      — next one. . .
   — second item. . .
```

\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 \labelwidthFB

Let’s define three lengths \listindentFB, \descindentFB and \labelwidthFB to customise lists’ horizontal indentations. They are given silly values here (-1 pt) in order to eventually enable their customisation in the preamble. They will get reasonnable defaults later when entering French (see \bbl@frenchlabelitems) unless they have been customised.

```
\newlength\listindentFB
\setlength{\listindentFB}{-1pt}
\newlength\descindentFB
\setlength{\descindentFB}{-1pt}
\newlength\labelwidthFB
\setlength{\labelwidthFB}{-1pt}
```

\FB@listHsettings \leftmarginFB

\FB@listHsettings holds the new horizontal settings chosen for French lists itemize and enumerate starting with version 2.6a. They are based on the look resquested in French for itemize-lists.

```
\newlength\leftmarginFB
\def\FB@listHsettings{%
   \leftmarginFB\labelwidthFB
   \advance\leftmarginFB \labelsep
   \bbl@for\FB@dp {1, 2, 3, 4, 5, 6}%
     {\csname leftmargin\romannumeral\FB@dp\endcsname \leftmarginFB}%
   \advance\leftmargini \listindentFB
   \leftmargin\csname leftmargin\ifnum\@listdepth=\@ne i\else
                                                    ii\fi\endcsname
}
```

\itemizeFB \FB@itemizesettings

New environment for French itemize-lists.

\FB@itemizesettings does two things: first suppress all vertical spaces including glue when option ReduceListSpacing is set, then set horizontal indentations according to \FB@listHsettings unless option ListOldLayout is true (compatibility with lists up to v. 2.5k).

```
\def\FB@itemizesettings{%
    \ifFBReduceListSpacing
      \setlength{\itemsep}{\z@}%
      \setlength{\parsep}{\z@}%
      \setlength{\topsep}{\z@}%
      \setlength{\partopsep}{\z@}%
      \@tempdima=\parskip
      \addtolength{\topsep}{-\@tempdima}%
      \addtolength{\partopsep}{\@tempdima}%
    \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 2ε classes (see `ltlists.dtx`), spaces are customised by `\FB@itemizesettings`.

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

```
\def\labelitemsFB{%
    \let\labelitemi\Frlabelitemi
    \let\labelitemii\Frlabelitemii
    \let\labelitemiii\Frlabelitemiii
    \let\labelitemiv\Frlabelitemiv
    \ifdim\labelwidthFB<\z@
      \settowidth{\labelwidthFB}{\FrenchLabelItem}%
    \fi
    \ifdim\listindentFB<\z@
      \ifdim\parindent=\z@
        \setlength{\listindentFB}{1.5em}%
      \else
        \setlength{\listindentFB}{\parindent}%
      \fi
    \fi
    \ifdim\descindentFB<\z@
      \setlength{\descindentFB}{\listindentFB}%
    \fi
}
```

`\enumerateFB` The definition of `\enumerateFB`, new to version 2.6a, follows the one of `\enumerate` in standard LaTeX 2ε 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\@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 length \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.

```
\def\descriptionFB{%
      \list{}{\FB@listHsettings
              \labelwidth\z@
              \itemindent-\leftmargin
              \ifnum\@listdepth=1
                \ifdim\descindentFB=\z@
                  \ifdim\listindentFB>\z@
                    \leftmargini\listindentFB
                    \leftmargin\leftmargini
                    \itemindent-\leftmargin
                  \fi
                \else
                  \advance\itemindent by \descindentFB
                \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{%
  \ifFBReduceListSpacing \let\list\listFB \fi
  \ifFBStandardItemizeEnv
  \else \let\itemize\itemizeFB \fi
  \ifFBStandardItemLabels
  \else \labelitemsFB \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. 56.

```
\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
}
\FB@addto{extras}{\bbl@frenchlistlayout}
```

2.12 French indentation of sections

\bbl@frenchindent
\bbl@nonfrenchindent

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.
We will need to save the value of the flag \if@afterindent 'AtBeginDocument' before eventually changing its value.

```
\def\bbl@frenchindent{%
  \ifFBGlobalLayoutFrench
  \else
    \babel@save\@afterindentfalse
  \fi
  \ifFBIndentFirst
    \let\@afterindentfalse\@afterindenttrue
    \@afterindenttrue
  \fi}
\def\bbl@nonfrenchindent{%
  \ifFBGlobalLayoutFrench
    \ifFBIndentFirst
      \@afterindenttrue
    \fi
  \fi}
\FB@addto{extras}{\bbl@frenchindent}
\FB@addto{noextras}{\bbl@nonfrenchindent}
```

2.13 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 2.10). 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{frenchb.ldf}%
                     {bigfoot package in use.\MessageBreak
                      frenchb 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}
\newlength\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 will be 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.

```
\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:

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

Special add-on for the `memoir` class: `\maketitle` redefines `\@makefntext` as `\makethanksmark` which is customised as follows 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 classes other than koma-script:

```
        \long\def\@makefntext#1{%
                  \ifFBFrenchFootnotes
                    \@makefntextFB{#1}%
                  \else
                    \@makefntextORI{#1}%
                  \fi}%
      \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 2.10) 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}
```

2.14 Clean up and exit

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

```
\FBclean@on@exit
\let\FB@llc\loadlocalcfg
\let\loadlocalcfg\@gobble
\ldf@finish\CurrentOption
\let\loadlocalcfg\FB@llc
```

3 Change History

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

v3.3a

General: Compatibility code for pre 2015/10/01 LaTeX release removed, see ltnews23.tex. 17

\captionsfrench: Commands \frenchpartfirst, \frenchpartsecond and \frenchpartnameord added. . . . 41

\FBguillspace: Skip \FBguillskip for LuaTeX replaced by toks \FBguillsp. 31

\FBthinspace: Skips \FBcolonskip and \FBthinskip replaced by toks \FBcolonsp and \FBthinsp. 16

\frenchsetup: \frenchbsetup is now an alias for \frenchsetup. . 47

Options INGuillSpace, ThinColonSPace no longer delayed AtBeginDocument. 47

\frquote: \FB@quotespace (kern), changed into \FB@guillspace. . 33

v3.2h

\@makefntextFB: With beamer.cls, add \llap to \@thefnmark for notes numbered over 99. 67

\bbl@frenchlistlayout: Execute \update@frenchlists only if GlobalLayoutFrench is false. Delete stuff for lists in \noextrasfrench. 65

\frenchsetup: Option GlobalLayoutFrench skipped when French is not the main language. 48

v3.2g

General: Add \boi to redefinitions for bookmarks. 59

Changed Unicode definition of \boi. 38

fontspec defines TU encoding now and no longer loads xunicode.sty. Test changed. 60

Issue a warning if beamerarticle.sty is loaded after babel. 47

\frenchsetup: Minimal list customisation when beamerarticle.sty is loaded. . . . 48

Warn when wrong values are provided to options EveryParGuill or EveryLineGuill. 52

\frquote: Default options of \frquote are no longer engine-dependent. 32

v3.2f

\DecimalMathComma: Fixed conflict with the icomma package. 39

v3.2e

General: Add missing redefinitions for \leftmarginv, \leftmarginvi. Suggested by J.F. Burnol. 62

\DecimalMathComma: \DecimalMathComma didn't work with LuaTeX. Fixed now. 39

v3.2d

\descriptionFB: Changed \listindentFB to \descindentFB which defaults to \listindentFB. \leftmargini reduced when \descindentFB is null. 64

v3.2c

General: New LuaTeX attribute \FB@spacing. 17

Newif \ifFB@spacing and new commands \FB@spacingon, \FB@spacingoff to control space tuning in French. 16

Switch \ifFB@spacing added to the four French shorthands. 27

\FB@xetex@punct@french: Switch \ifFB@spacing added to all \XeTeXinterchartoks commands. 25

\FBthinspace: Change .16667em to .5\fontdimen2\font to get in XeTeX and pdfTeX the same spacing as in LuaTeX. 16

\frenchsetup: Add a warning about options og/fg for old XeTeX or LuaTeX engines requiring active characters. 53

\NoAutoSpacing: New definition based on \FB@spacing@off common to all engines. 30