The tutodoc class - Tutorial-style documentation

Christophe BAL

Sep 28, 2024 - Version 1.4.0

Abstract

The tutodoc class 1 is used by its author to semantically produce documentation of LATEX packages and classes in a tutorial style 2 using a sober rendering for reading on screen.

Remark: this documentation is also available in French.

¹The name comes from " $tuto \cdot rial - type \ doc \cdot umentation$ ".

²The idea is to produce an efficient PDF file that can be browsed for one-off needs. This is generally what is expected of coding documentation.

Contents

I.	General formatting imposed	3
	1. Page geometry	3
	2. Title and table of contents	3
	3. Dynamic links	4
II.	What language is used by the tutodoc class?	4
III.	What does that mean in "English"?	4
IV.	Highlighting content	5
	1. Content in the reading flow	5
	i. Examples	5
	ii. Some remarks	5
	2. Flashy content	6
	i. A tip	6
	ii. Informative note	6
	iii. Something important	6
	iv. Caution about a delicate point	7
	v. Warning of danger	7
V.	Specify packages, classes, macros or environments	7
VI.	Origin of a prefix or suffix	8
VII.	A real-life rendering	8
	1. With a colored stripe	8
	2. Without a colour strip	9
	3. By importing the LATEX code	10
VIII.	Use cases in \LaTeX	10
	1. "Inline" codes	11
	2. Directly typed codes	11
	3. Imported codes	12
	4. Imported codes put into practice	13
IX.	Indicate changes	14
	1. When?	14
	2. What's new?	15
Χ.	Ornaments	16
XI.	Contribute	17
	1. Complete the translations	17
	i. The fr and en folders	17
	ii. The changes folder	17
	iii. The status folder	17
	iv. The README.md and LICENSE.txt files	17
	v. New translations	17
	2. Improving the source code	18
XII.	History	19

I. General formatting imposed

1. Page geometry

The geometry package is loaded with the following settings.

2. Title and table of contents

The titlesec and tocbasic packages are set as follows.

```
\RequirePackage[raggedright]{titlesec}
\ifcsundef{chapter}%
          {}%
          {\renewcommand\thechapter{\Alph{chapter}.}}
\renewcommand\thesection{\Roman{section}.}
\renewcommand\thesubsection{\arabic{subsection}.}
\renewcommand\thesubsubsection{\roman{subsubsection}.}
\titleformat{\paragraph}[hang] %
            {\normalfont\normalsize\bfseries}%
            {\theparagraph}{1em}%
            {}
\titlespacing*{\paragraph}%
              {0pt}%
              {3.25ex plus 1ex minus .2ex}%
              \{0.5em\}
% Source
* https://tex.stackexchange.com/a/558025/6880
\DeclareTOCStyleEntries[
 raggedentrytext,
 linefill = \hfill,
 indent = Opt,
 dynindent,
 numwidth = Opt,
 numsep = 1ex,
 dynnumwidth
]{tocline}{
 chapter,
 section,
 subsection,
 subsubsection,
 paragraph,
 subparagraph
\DeclareTOCStyleEntry[indentfollows = chapter]{tocline}{section}
```

3. Dynamic links

The hyperref package is imported behind the scenes with the settings below.

```
\hypersetup{
  colorlinks,
  citecolor = orange!75!black,
  filecolor = orange!75!black,
  linkcolor = orange!75!black,
  urlcolor = orange!75!black
}
```

II. What language is used by the tutodoc class?

This documentation loads the babel package via \usepackage[english] {babel}. As a result, the tutodoc class identifies en as the main language used by babel.³ As this language is included in the list of languages taken into account, see below, the tutodoc class will produce the expected effects. «LANG-SUPPORTED»

& Caution.

If the choice of main language is not made in the preamble, the mechanism used will fail with unintended side effects (see warning that follows).

& Warning.

When a language is not supported by tutodoc, a warning message is issued, and English is selected as the language for tutodoc.

i Note.

The mechanism used should be compatible with the polyglossia package.

III. What does that mean in "English"?

The macro \tdocinEN and its starred version are useless for English speakers because they have the following effects.

```
Cool and top stand for \tdocinEN*{cool} and \tdocinEN{top}.

Cool and top stand for "cool" and "top" in english.
```

The macro \tdocinEN and its starred version are based on \tdocquote: for example, "semantic" is obtained via tdocquote{semantic}.

1 Note.

As the text "in English" is translated into the language detected by tutodoc, the macro \tdocinen and its starred version become useful for non-English speakers.

³Technically, we use \BCPdata{language} which returns a language in short format.

IV. Highlighting content



The environments presented in this section ^a add a short title indicating the type of information provided. This short text will always be translated into the language detected by the tutodoc class.

 a The formatting comes from the keytheorems package.

1. Content in the reading flow

Important

All the environments presented in this section share the same counter.

i. Examples

Numbered examples, if required, are indicated via \begin{tdocexa} . . . \end{tdocexa}, which offers an optional argument for adding a mini-title. Here are two possible uses.

\begin{tdocexa}
An example...
\end{tdocexa}

Example IV.1. An example...

\begin{tdocexa} [Mini title]
Useful?
\end{tdocexa}

Important.

The numbering of the examples is reset to zero as soon as a section with a level at least equal to a \section is opened.

• Tip.

It can sometimes be useful to return to the line at the start of the content. The code below shows how to proceed (this trick also applies to the tdocrem environment presented next). Note in passing that the numbering follows that of the previous example as desired.

ii. Some remarks

Everything happens via \begin{tdocrem} ...\end{tdocrem}, which works identically to the tdocexa environment, as shown in the following example.

```
begin{tdocrem}
    Just one remark...
\end{tdocrem}

kbegin{tdocrem}
    Remark IV.4. Just one remark...

kbegin{tdocrem}
    Remark IV.5. Another?

kend{tdocrem}

Remark IV.6 (Mini title). Useful?

kend{tdocrem}

useful?

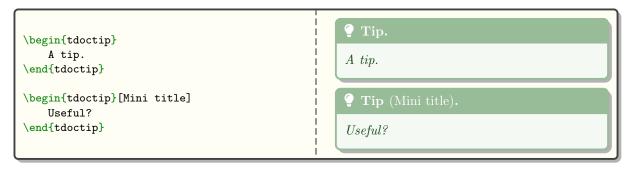
lend{tdocrem}
```

2. Flashy content



i. A tip

The tdoctip environment is used to give tips. Here's how to use it.

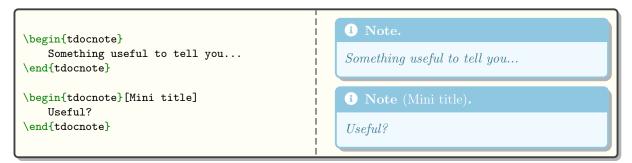


i Note.

Colors are obtained via the expandable macros \t docbackcolor and \t docdarkcolor. For further information, please refer to the end of the section 1. page 8.

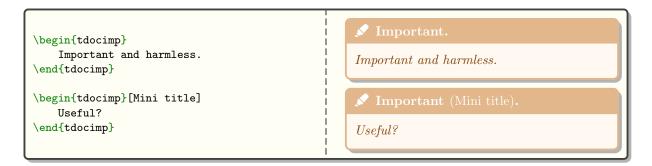
ii. Informative note

The tdocnote environment is used to highlight useful information. Here's how to use it.



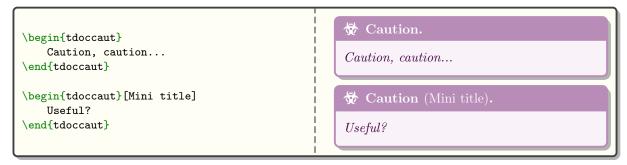
iii. Something important

The tdocimp environment is used to indicate something important but harmless.



iv. Caution about a delicate point

The tdoccaut environment is used to indicate a delicate point to the user. Here's how to use it.



v. Warning of danger

The tdocwarn environment is used to warn the user of a trap to avoid. Here's how to use it.

```
\begin{tdocwarn}
Avoid the dangers...
\end{tdocwarn}

\begin{tdocwarn} [Mini title]
Useful?
\end{tdocwarn}

\text{Useful?}
```

V. Specify packages, classes, macros or environments

Here's what you can type semantically.

```
myclass is for...
\tdoccls{myclass} is for...
                                          11
                                                   mypackage is for...
\tdocpack{mypackage} is for...
                                          //
                                                   \onemacro is for...
\tdocmacro{onemacro} is for...
\tdocenv{env} produces...
                                                   \begin{env} ...\end{env} produces...
\tdocenv[{[opt1]<opt2>}]{env}
                                                   \begin{env}[opt1]<opt2>...\end{env}
Just \tdocenv*{env}...
                                                   Just env...
Finally \tdocenv*[{[opt1]<opt2>}]{env}...
                                                   Finally env...
```

Remark V.1. Unlike \tdocinlatex, \tdocenv and \tdocenv* macros don't color the text they produce. In addition, \tdocenv{monenv} produces \begin{monenv}...\end{monenv} with spaces to allow line breaks if required.



The optional argument of the \t docenv macro is copied and pasted ^a when rendering. This may sometimes require the use of protective braces, as in the example above.

```
<sup>a</sup>Remember that almost anything is possible from now on.
```

VI. Origin of a prefix or suffix

To explain the names chosen, there is nothing like indicating and explaining the short prefixes and suffixes used. This is easily done as follows.

```
\tdocpre{sup} relates to...
sup relates to...

\tdocprewhy{sup.erbe} means...
sup·erbe means...

\emph{\tdocprewhy{sup.er} for...}
sup·er for...
```

Remark VI.1. The choice of a full stop to split a word allows words with a hyphen to be used, as in \tdocprewhy{bric.k-breaker} which gives bric·k-breaker.

VII. A real-life rendering

It is sometimes useful to render code directly in the documentation. This type of rendering must be dissociable from the explanatory text.

1. With a colored stripe

Example VII.1 (With default text). It can be useful to show a real rendering directly in a document. ⁴ This is done via \begin{tdocshowcase} ... \end{tdocshowcase} as follows.

```
| begin{tdocshowcase}
| bfseries A bit of code \LaTeX.
| bigskip
| emph{\large End of the awful demo.}
| end{tdocshowcase}
```

The result is the following rendering. ⁵

Start of the real output

A bit of code LATEX.

End of the awful demo.

End of the real output

Remark VII.2. See the section 4. on page 13 to easily obtain code followed by its actual rendering as in the previous example.

i Note.

The explanatory texts adapt to the language detected by tutodoc.

Example VII.3 (Change the default colour and/or text).

⁴Typically when making a demo.

⁵Behind the scenes, the strip is created effortlessly using the clrstrip package.

This will produce the following.

i Note.

You've probably noticed that red is used as a base to obtain the colors used.

- The background color is provided by \tdocbackcolor.
- The color of titles and lines is provided by \tdocdarkcolor.

These expandable macros accept the following codes.

You also have to know that behind the scene, the \tdocruler macro is used.

\tdocruler[red]{A decorated pseudo-title}

A decorated pseudo-title

& Warning.

With the default settings, if the code to be formatted begins with an opening bracket, use \string as in the following example.

```
\begin{tdocshowcase}
  \string[This works...]
\end{tdocshowcase}
```

This will produce the following.

```
[This works...]

End of the real output
```

2. Without a colour strip

The rendering of \begin{tdocshowcase} ...\end{tdocshowcase} with a coloured strip may not be suitable, or sometimes may not be acceptable despite the work done by clrstrip. It is possible not to use a coloured strip, as we will see straight away.

Example VII.4. The use of \begin{tdocshowcase} [nostripe] . . . \end{tdocshowcase} indicate to not use clrstrip. Here is an example.

```
\begin{tdocshowcase}[nostripe]
    \end{tdocshowcase}
This will produce the following.
                           ■ Start of the real output ■
■ End of the real output ■
Example VII.5 (Change the default colour and/or text).
 \begin{tdocshowcase} [nostripe,
                before = My beginning,
                after = My end,
                color = green]
    \end{tdocshowcase}
This will produce the following.
                              ■ My beginning
■ My end ■
   By importing the LATEX code
To obtain renderings by importing the code from an external file, instead of typing it, simply use
the \tdocshowcaseinput macro whose option uses the syntax of that of \begin{tdocshowcase} ...
```

\end{tdocshowcase} and the mandatory argument corresponds to the path of the file.

Example VII.6. The following was obtained via \tdocshowcaseinput {external.tex}.

Start of the real output	
$Blablobli,\ blablobli,\ blablobli,\ blablobli,\ blablobli,\ blablobli$	
End of the real output	
As for \toolongle the colour change shown below.	
Start of the real output	
Blablobli, blablobli, blablobli, blablobli, blablobli, blablobli	
End of the real output	

VIII. Use cases in LATEX

Documenting a package or class is best done through use cases showing both the code and the corresponding result. ⁶

⁶Code is formatted using the minted package.

1. "Inline" codes

The \tdocinlatex macro ⁷ can be used to type inline code in a similar way to \verb or like a standard macro (see brace management in the last case below). Here are some examples.

```
1: \tdocinlatex|\$a^b = c\$| \\ 2: \tdocinlatex+\tdocinlatex|\$a^b = c\$|+ \\ 3: \tdocinlatex\{\tdocinlatex\{\$a^b = c\$}\} \\ 3: \tdocinlatex\{\$a^b = c\$}\
```

1 Note.

The \tdocinlatex macro can be used in a footnote: see below. a In addition, a background color is deliberately used to subtly highlight the codes \LaTeX.

a\$minted = TOP\$ has been typed \tdocinlatex+\$minted = TOP\$+ in this footnote...

2. Directly typed codes

Example VIII.1 (Side by side). Using \begin{tdoclatex}[sbs]...\end{tdoclatex}, we can display a code and its rendering side by side. Consider the following code.

```
\begin{tdoclatex}[sbs]

$A = B + C$
\end{tdoclatex}
```

This will produce the following.

```
\$A = B + C\$
A = B + C
```

Example VIII.2 (Following). $\begin{tabclatex} begin{tabclatex} ... \end{tabclatex} produces the following result, which corresponds to the default option std. }$

```
A = B + C
A = B + C
```

Example VIII.3 (Just the code). Via \begin{tdoclatex}[code] ... \end{tdoclatex}, we'll just get the code as shown below.

```
$A = B + C$
```

& Warning.

With default formatting, if the code begins with an opening bracket, the default option must be explicitly indicated. Consider the following code.

```
\begin{tdoclatex}[std]
[Strange... Or not!]
\end{tdoclatex}
```

This will produce the following.

⁷The name of the macro \tdocinlatex comes from "in·line partial TEX".

⁸std refers to the "standard" behaviour of tcolorbox in relation to the minted library.

```
[Strange... Or not!]

Another method is to use the \string primitive. Consider the following code.

\[
\begin{tdoclatex} \string[Strange... Or not!] \end{tdoclatex}
\]

This will produce the following.

[Strange... Or not!]

[Strange... Or not!]
```

3. Imported codes

For the following codes, consider a file with the relative path examples-listing-xyz.tex, and with the following contents.

```
% Just one demo.
$x y z = 1$
```

The \tdoclatexinput macro, shown below, expects the path of a file and offers the same options as the tdoclatex environment.

Example VIII.4 (Side by side).

```
\to clatexinput[sbs]{examples-listing-xyz.tex}
```

This produces the following layout.

```
% Just one demo.
$x y z = 1$ xyz = 1
```

Example VIII.5 (Following).

This produces the following formatting where the default option is std.

```
% \ Just \ one \ demo.
\$x \ y \ z = 1\$
xyz = 1
```

Example VIII.6 (Just the code).

```
\to clatexinput[code]{examples-listing-xyz.tex}
```

 $This\ produces\ the\ following\ layout.$

```
% Just one demo.
$x y z = 1$
```

4. Imported codes put into practice

Example VIII.7 (Showcase). The following comes from \tdoclatexshow{examples-listing-xyz.tex}.

Start of the rendering in this doc.

"Just one demo.
"x y z = 1\$

This gives:

Start of the real output

End of the real output

■ End of rendering in this doc.

i Note.

xyz = 1

xyz = 1

The default texts take into account the language detected by tutodoc.

Example VIII.8 (Changing the explanatory text). Using the key explain, you can use custom text. Thus, tdoclatexshow[explain = Here is the actual rendering.] {examples-listing-xyz.tex} will produce the following.

% Just one demo.
\$x y z = 1\$

Here is the actual rendering.

End of the real output

Start of the real output

Example VIII.9 (The options available). In addition to the explanatory text, it is also possible to use all the options of tdocshowcase environment, see VII. page 8. Here is an example to illustrate this.

\tdoclatexshow[explain = What comes next is colourful...,
before = Rendering below.,
after = Finished rendering.,
color = orange]
{examples-listing-xyz.tex}

This will produce the following.

Start of the rendering in this doc.

```
% Just one demo.
$x y z = 1$
```

What comes next is colourful...



■ End of rendering in this doc.

IX. Indicate changes

To make it easier to monitor a project, it is essential to provide a history indicating the changes made when a new version is published.

1. When?

You can either date something, or version it, in which case the version number can be dated.

Example IX.1 (Dating new products). The \tdocdate macro is used to indicate a date in the margin, as in the following example.

This gives:

2023-09-24

2020-05-08

Example IX.2 (Versioning new features, possibly with a date). Associating a version number with a new feature is done using the \tag{tdocversion} macro, with the colour and date being optional arguments.

■ End of the real output ■

This gives:

■ Start of the real output ■

10.2.0-beta 2023-12-01

10.2.0-alpha

■ End of the real output ■

🖍 Important.

- 1. The \tdocdate and \tdocversion macros require two compilations.
- 2. The final rendering of the dates takes into account the language detected by tutodoc: for example, if French is selected, the dates will be displayed in the format DD/MM/YYYY.

🙎 Warning.

Only the use of the digital format YYYY-MM-DD is verified, ^a and this is a choice! Why? Quite simply because dating and versioning explanations should be done semi-automatically to avoid any human bugs.

^aTechnically, checking the validity of a date using I⁴TEX3 presents no difficulty.

2. What's new?

tutodoc offers the macro \tdocstartproj and different environments to indicate quickly and clearly what has been done during the latest changes.⁹

1 Note.

For icons, see the note at the beginning of the section 2. page 6.

Example IX.3 (Just for the very first version).

\tdocstartproj{1st version of the project.} \ \frac{1}{3} \ 1st version of the project.

Example IX.4 (For new features).

\begin{tdocnew}	♦ New.
\item Info 1\item Info 2	• Info 1
\end{tdocnew}	• Info 2

Example IX.5 (For updates).

\begin{tdocupdate}	₩ UPDATE.
\item Info 1 \item Info 2	• Info 1
\end{tdocupdate}	• Info 2

⁹The user doesn't need all the technical details.

Example IX.6 (For breaks).

\begin{tdocbreak}	₿ Break.
\item Info 1\item Info 2	• Info 1
\end{tdocbreak}	• Info 2

Example IX.7 (For problems).

\begin{tdocprob}	♦ PROBLEM.
\item Info 1\item Info 2	• Info 1
$\ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ $	• Info 2

Example IX.8 (For fixes).

\begin{tdocfix}	⊁ Fix.
\item Info 1 \item Info 2	• Info 1
\end{tdocfix}	• Info 2

Example IX.9 (Selectable themes with an icon).

\begin{tdoctopic}{Don't look}[\faEyeSlash] % An icon from fontawesome5.	№ Don't look.
\item Info 1	• Info 1
\item Info 2 \end{tdoctopic}	• Info 2

Example IX.10 (Selectable themes without icons).

\begin{tdoctopic}{End of icons}	END OF ICONS.
\item Info 1 \item Info 2	• Info 1
\end{tdoctopic}	• Info 2

X. Ornaments

Let's finish this documentation with a small formatting tool that is very useful.

Bla, bla, bla \tdocsep % Practical for demarcation.	Bla, bla, bla
This works with enumerations. begin{itemize} item Underline. bend{itemize}	This works with enumerations. • Underline.
\tdocsep % Uniform behaviour. Ble, ble, ble	Ble, ble, ble

XI. Contribute



You don't need to be a coder to take part in translations, including those that are useful for the running of tutodoc.

1. Complete the translations

1 Note.

The author of tutodoc manages the French and English versions of the translations.

☆ Caution.

Although we're going to explain how to translate the documentation, it doesn't seem relevant to do so, as English should suffice these days. a

^aThe existence of a French version is simply a consequence of the native language of the author of tutodoc.

The translations are roughly organized as in figure 1 where only the folders important for the translations have been "opened". ¹⁰ A little further down, the v. section explains how to add new translations.

i. The fr and en folders

These two folders, managed by the author of tutodoc, have the same organization; they contain files that are easy to translate even if you're not a coder.

ii. The changes folder

This folder is a communication tool where important changes are indicated without dwelling on minor modifications specific to one or more translations.

iii. The status folder

This folder is used to keep track of translations from the project's point of view. Everything is done via well-commented YAML files, readable by a non-coder.

en doc fr status en api.yaml manual.yaml fr README.md LICENSE.txt

🖿 translate

changes

Figure 1: Simplified view of the translation folder

iv. The README.md and LICENSE.txt files

The LICENSE.txt file is aptly named, while the README.md file takes up in English the important points of what is said in this section about new translations.

v. New translations

Mark Important.

The api folder contains translations relating to the functionalities of tutodoc. Here you'll find TXT files for editing with a text or code editor, but not with a word processor. The contents of these files use commented lines in English to explain what tutodoc will do; these lines begin with // . Here's an extract from such a file.

¹⁰This was the organization on October 5, 2024, but it's still relevant today.

```
// #1: year in format YYYY like 2023.
// #2: month in format MM like 04.
// #3: day in format DD like 29.
date = #1-#2-#3

// #1: the idea is to produce one text like
// "this word means #1 in english".
in_EN = #1 in english
```

The translations are made after each sign = without touching what comes before, as this initial piece is used internally by the <math>tutodoccode.

1 Note.

The doc folder is reserved for documentation. It contains files of type TEX that can be compiled directly for real-time validation of translations.

& Warning.

Only start from one of the fr and en folders, as these are the responsibility of the tutodoc author.

Let's say you want to add support for Italian from files written in English. 11

Method 1: git.

- 1. Obtain the entire project folder via https://github.com/bc-tools/for-latex/tree/tutodoc. Do not use the main branch, which is used to freeze the latest stable versions of projects in the single https://github.com/bc-tools/for-latex repository,.
- 2. In the tutodoc/contrib/translate folder, create a it copy of the en folder, with the short name of the language documented in the page "*HETF language tag*" from Wikipedia.
- 3. Once the translation is complete in the it folder, you'll need to communicate it via https://github.com/bc-tools/for-latex/tree/tutodoc using a classic git push.

Method 2: communicate by e-mail.

- By e-mail with the subject "tutodoc CONTRIB en FOR italian", request a version of the English translations (note the use of the English name for the new language). Be sure to respect the subject of the e-mail, as the author of tutodoc automates the pre-processing of this type of e-mail.
- 2. You will receive a folder named italian containing the English version of the latest translations. This folder will be the place for your contribution.
- 3. Once the translation is complete, you will need to compress your italian file in zip or rar format before sending it by e-mail with the subject "tutodoc CONTRIB italian".

2. Improving the source code

Participation as a coder is made via the https://github.com/bc-tools/for-latex/tree/tutodoc repository corresponding to the tutodoc development branch. Do not use the main branch, which is used to freeze the latest stable versions of projects in the single https://github.com/bc-tools/for-latex repository.

¹¹As mentioned above, there is no real need for the doc documentation folder.

💉 Important.

If you want to participate in tutodoc you'll need to use the LATEX3 programming paradigm.

XII. History

1.4.0 2024-09-28

P BREAK.

- The tdoccaution environment has been renamed tdoccaut for simplified input.
- Content highlighting: examples and remarks, indicated via the tdocexa and tdocrem environments, are always numbered via a common counter.
- The unused macro \tdocxspace has been deleted.

NEW.

- Change log: the \tdocstartproj macro is used to manage the case of the first public version.
- Code factorization: the \tdocicon macro is responsible for adding icons in front of text.

Z UPDATE.

- Colors: the \tdocdarkcolor and \tdoclightcolor macros offer an optional argument.
 - 1. \tdocdarkcolor: the amount of color in relation to black can be optionally defined.
 - 2. \tdoclightcolor: the transparency rate can be optionally defined.
- Content highlighting: reduced space around content in colored frames.
- Versioning: better vertical spacing thanks to \vphantom.

1.3.1 2024-09-26

NEW.

• Star version of \tdocenv to display only the environment name.

1.3.0 2024-09-25

• Problem.

• Version 3 of minted cannot be used for the moment as it contains bugs: see https://github.com/gpoore/minted/issues/401. We therefore force the use of version 2 of minted.

P BREAK.

• The tdocimportant environment has been renamed tdocimp for simplified input.

NEW.

- Change log: proposed environments use icons.
- Content highlighting: colored frames with icons are proposed for the following environments.
 - 1. tdoccaution
- 3. tdocnote

5. tdocwarn

 $2.\ {\tt tdocimp}$

4. tdoctip

1.2.0-a 2024-08-23

W UPDATE.

- \tdocversion
 - 1. The version number is above the date.
 - 2. The spacing is better managed when the date is absent.

Fix.

 \bullet Content highlighting: the French translations of "caution" and "danger" were incorrect.

1.1.0 2024-01-06

NEW.

- Change log: two new environments.
 - 1. \begin{tdocbreak} ...\end{tdocbreak} for breaking changes which are not backward compatible.
 - 2. $\begin{tdocprob} \dots \end{tdocprob} for identified problems.$

 \bullet \tdocinlatex: a light yellow is used as the background color.

1.0.1 2023-12-08

Fix.

- \tdocenv: spacing is now correct, even if the babel package is not loaded with the French language.

1.0.0 2023-11-29

\$ First public version of the project.