# The tutodoc package - Tutorial-style documentation

## Christophe BAL

Sep 26, 2024 - Version 1.3.1

#### Abstract

The tutodoc package <sup>1</sup> is used by its author to semantically produce documentation of LATEX packages and classes in a tutorial style <sup>2</sup>, and with a sober rendering for reading on screen.

Two important points to note.

- This package imposes a formatting style. In the not-too-distant future, tutodoc will probably be split into a class and a package.
- This documentation is also available in French.

<sup>&</sup>lt;sup>1</sup>The name comes from " $tuto \cdot rial - type \ doc \cdot umentation$ ".

<sup>&</sup>lt;sup>2</sup>The 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.	Select language when loading package	4
III.	What does that mean in "English"?	4
IV.	Highlighting content	4
	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
	3. Caution about a delicate point	7
	i. 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 coloured 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
X.	Ornaments	16
XI.	History	17

## 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. Select language when loading package

By default, tutodoc is set for English, but it is possible to change the language: for example, a French documentation will use \usepackage[lang = french] {tutodoc} . For the moment, we only have the following two choices.

- 1. english is the default value.
- 2. french



Language names are those suggested by the babel 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}.

#### i Note.

As the text "in English" is translated into the language indicated when tutodoc is imported, the macro \tdocineN and its starred version become useful for non-English speakers.

## IV. Highlighting content



The environments presented in this section <sup>a</sup> add a short title indicating the type of information provided. This short text will always be translated into the language indicated when the tutodoc package is loaded.

 $^a$ The formatting comes from the keytheorems package.

### 1. Content in the reading flow

#### i. Examples

Numbered or unnumbered examples can be indicated using the \begin{tdocexa} ...\end{tdocexa} environment, which offers two optional arguments.

- 1. The 1<sup>st</sup> argument between brackets <...> can take the values **nb** to number, which is the default setting, and **nonb** to not number.
- 2. The  $2^{\rm nd}$  argument in square brackets [...] is used to add a mini-title..

Here are some possible uses.

```
Bla, bla, bla...
\begin{tdocexa}
    Ble, ble, ble...
\end{tdocexa}
                                                    Bla, bla, bla...
\begin{tdocexa} [Wonderful]
                                                    Example 1. Ble, ble, ble...
    Bli, bli, bli...
                                                    Example 2 (Wonderful). Bli, bli, bli...
\end{tdocexa}
                                                    Example. Blo, blo, blo...
\begin{tdocexa}<nonb>
    Blo, blo, blo...
                                                    Example (Superb). Blu, blu, blu...
\end{tdocexa}
\begin{tdocexa}<nonb>[Superb]
    Blu, blu, blu...
\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 \subsubsection 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.

```
\begin{tdocexa}
\leavevmode
\begin{enumerate}
    \item Point 1.

    \item Point 2.
    \end{enumerate}
\end{tdocexa}
Example 3.

1. Point 1.

2. Point 2.
```

### ii. Some remarks

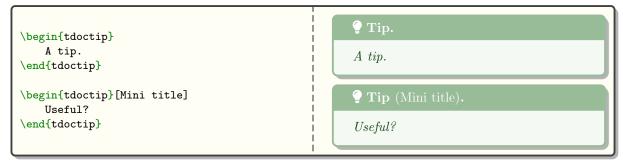
Everything happens via the \begin{tdocrem} ...\end{tdocrem} environment, as in the following example.

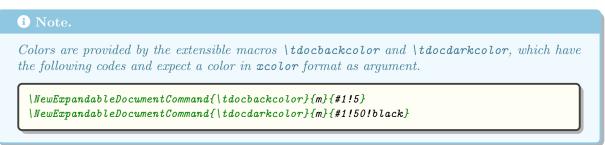
```
\begin{tdocrem}
    Just one remark...
\end{tdocrem}
    Remark. Just one remark...
\begin{tdocrem}[Mini title]
    Useful?
\end{tdocrem}
Remark (Mini title). Useful?
```

### 2. Flashy content

### i. A tip

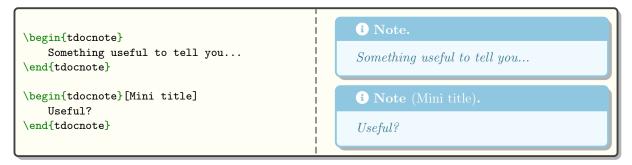
The \begin{tdoctip} ...\end{tdoctip} environment is used to give tips. Here's how to use it.





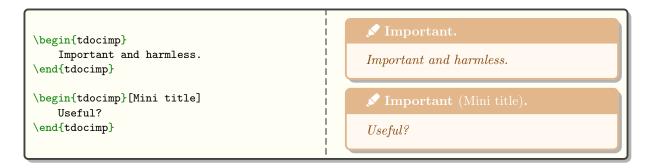
### ii. Informative note

The \begin{tdocnote} ...\end{tdocnote} environment is used to highlight useful information. Here's how to use it.



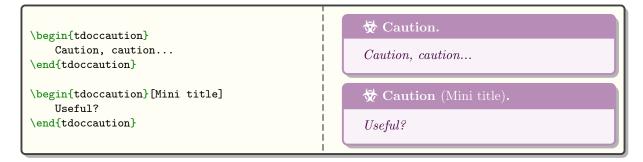
### iii. Something important

The \begin{tdocimp} ...\end{tdocimp} environment is used to indicate something important but harmless.



## 3. Caution about a delicate point

The \begin{tdoccaution} ...\end{tdoccaution} environment is used to indicate a delicate point to the user. Here's how to use it.



### i. Warning of danger

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



## V. Specify packages, classes, macros or environments

Here's what you can type semantically.

```
\tdoccls{myclass} is for...
\tdocpack{mypackage} is for...
\tdocmacro{onemacro} is for...
\tdocmacro{onemacro} is for...
\tdocenv{env} produces...
\docenv*{env} ...

Just \tdocenv*{env}...
\tdocenv*{env}...

\tdocenv[{[opt1] < opt2>}]{env}...
\finally \tdocenv*[{[opt1] < opt2>}]{env}...

myclass is for...

myclass is for...

homemacro is for...
\text{begin{env} ...\end{env} produces...}

Just env...
\text{begin{env} [opt1] < opt2> ...\end{env}
Finally env...}

Finally \tdocenv*[{[opt1] < opt2>}]{env}...
\text{finally env...}
```

Remark. 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.

## & Warning.

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

## 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...

\tdocprewhy{sup.erbe} means...

\sup relates to...

\sup \cerbe means...

\sup \cer for...

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

Remark. 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 coloured stripe

Example 1 (With default text). It can be useful to show a real rendering directly in a document. <sup>3</sup> 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. <sup>4</sup>

Start of the real output

A bit of code LATEX.

End of the awful demo.

End of the real output

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

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

<sup>&</sup>lt;sup>3</sup>Typically when making a demo.

<sup>&</sup>lt;sup>4</sup>Behind the scenes, the strip is created effortlessly using the clrstrip package.

### i Note.

The explanatory texts adapt to the language chosen when tutodoc is loaded.

Example 2 (Change the default colour and/or text).

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 have a single argument, the chosen color, and accept the following codes.

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

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

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 1. 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 2 (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
```

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.** The following was obtained via \tdocshowcaseinput{external.tex}.

		Start 0	, the real output		
Blablob	li, blablobli, blablobli, b	blablobli, blablobli, i	blablobli		
		End of	the real output		_
As for	$\ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ $	:[color = orange]	{external.tex} ,	this will produce t	$the\ colour$
shown	below.				

change

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.  $^5$ 

#### & Caution.

Version 3 of minted cannot be used at the moment, as it contains bugs: see <a href="https://github.com/gpoore/minted/issues/401">https://github.com/gpoore/minted/issues/401</a>. We therefore force the use of version 2 of minted.

 $<sup>^5\</sup>mathrm{Code}$  is formatted using the minted package.

### 1. "Inline" codes

The \tdocinlatex macro <sup>6</sup> can be used to type inline code in a similar way to \verb. Here are some examples.

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

2: \tdocinlatex+\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 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 2 (Following). \begin{tdoclatex} ... \end{tdoclatex} produces the following result, which corresponds to the default option std . 7

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

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

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

## & 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.

<sup>&</sup>lt;sup>6</sup>The name of the macro  $\t$ docinlatex comes from "in-line  $\t$ T $_{E}$ X".

<sup>&</sup>lt;sup>7</sup> std 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 \begin{tdoclatex} ...\end{tdoclatex} environment.

Example 1 (Side by side).

```
\to the description of the latest and the latest angles of the latest
```

This produces the following layout.

```
% Just one demo.

$x y z = 1$

xyz = 1
```

Example 2 (Following).

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

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

Example 3 (Just the code).

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

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

```
% Just one demo.
xyz = 1
```

## Imported codes put into practice

Example 1 (Showcase). The following comes from \tdoclatexshow{examples-listing-xyz.tex}.

Start of the rendering in this doc.

% Just one demo. xyz = 1

This gives:

■ Start of the real output ■

xyz = 1

■ End of the real output ■

■ End of rendering in this doc. ■

i Note.

The default texts take into account the language chosen when loading the package tutodoc.

Example 2 (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.

Start of the rendering in this doc.

% Just one demo. xyz = 1

Here is the actual rendering.

Start of the real output

xyz = 1

■ End of the real output ■

**Example 3** (The options available). In addition to the explanatory text, it is also possible to use all

■ End of rendering in this doc. ■

the options of \begin{tdocshowcase}...\end{tdocshowcase}, see VII. page 8. Here is an example to illustrate this.

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

This will produce the following.

Start of the rendering in this doc.

% Just one demo. xyz = 1

What comes next is colourful...

	Rendering below.
xyz = 1	
	Finished rendering.
	End of rendering in this doc.

IX. Indicate changes

To make it easier to monitor a package, 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 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 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 specified when loading the package 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.

<sup>a</sup>Technically, checking the validity of a date using IATEX3 presents no difficulty.

#### 2. What's new?

 ${\tt tutodoc} \ {\tt offers} \ {\tt different} \ {\tt environments} \ {\tt to} \ {\tt indicate} \ {\tt quickly} \ {\tt and} \ {\tt clearly} \ {\tt what} \ {\tt has} \ {\tt been} \ {\tt done} \ {\tt during} \ {\tt the} \ {\tt latest} \ {\tt changes.}^8$ 

Example 1 (For new features).

```
\begin{tdocnew}
\item Info 1...
\item Info 2...
\end{tdocnew}

NEW.

• Info 1...

• Info 2...
```

### Example 2 (For updates).

\begin{tdocupdate}	<b>₩</b> UPDATE.
\item Info 1 \item Info 2	• Info 1
\end{tdocupdate}	• Info 2

## Example 3 (For breaks).

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

<sup>&</sup>lt;sup>8</sup>The user doesn't need all the technical details.

### Example 4 (For problems).

\begin{tdocprob}	O Problem.
\item Info 1 \item Info 2	• Info 1
\end{tdocprob}	• Info 2

### Example 5 (For fixes).

$oxed{egin{content} oxed{begin\{tdocfix}}$	FIX.
\item Info 1 \item Info 2	• Info 1
$\ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ $	• Info 2

## Example 6 (Selectable themes with an icon).

## Example 7 (Selectable themes without icons).

```
\begin{tdoctopic}{End of icons}

% This is where the point needs to be put.
\item Info 1...
\item Info 2...
\end{tdoctopic}

END OF ICONS.

• Info 1...

• Info 2...
```

## X. Ornaments

Let's finish this documentation with a few small formatting tools that can be very useful.

```
Bla, bla, bla...
\tdocsep % Practical for demarcation.
This works with enumerations.
                                                     Bla, bla, bla...
\begin{itemize}
    \item Underline.
                                                     This works with enumerations.
    \item Something else useful.
\end{itemize}
                                                         • Underline.
\tdocsep % Uniform behaviour.
                                                         • Something else useful.
Ble, ble, ble...
                                                     Ble, ble, ble...
Bli, bli, bli...
                                                     Bli, bli, bli...
\tdocxspace % Subtle space
                                                     Blo, blo, blo...
            % but useful.
                                                     Blu, blu, blu...
Blo, blo, blo...
Blu, blu, blu...
```

## XI. History

1.3.1 2024-09-26

## New.

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

1.3.0 2024-09-25

#### **O** Problem.

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

#### 🎖 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. tdocimp

4. tdoctip

1.2.0-a 2024-08-23

## 🏿 UPDATE.

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

### Fix.

• 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.  $\begin{tdocprob} \dots \end{tdocprob} for identified problems.$
- \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.
- \begin{tdocshowcase} [nostripe] ... \end{tdocshowcase}: page breaks around "framing" lines should be rare from now on.

1.0.0 2023-11-29

First public version of the project.