Le package `tutodoc` - Documentation de type tutoriel

Christophe BAL

29 Nov. 2023 - Version 1.0.0

Abstract

The `tutodoc` package [1] is used by its author to semantically produce documentation of LaTeX packages and classes in a tutorial style [2], and with a sober rendering for reading on screen.

Note. *This package imposes a formatting style. In the not-too-distant future,* ***`tutodoc`*** *will probably be split into a class and a package.*

[1] The name comes from "***`tuto`*·*rial-type* ***`doc`*·*umentation***".

[2] 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

1. Page geometry

The `geometry` package is loaded with the following settings.

```
\RequirePackage[
  top            = 2.5cm,
  bottom         = 2.5cm,
  left           = 2.5cm,
  right          = 2.5cm,
  marginparwidth = 2cm,
  marginparsep   = 2mm,
  heightrounded
]{geometry}
```

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   = 0pt,
  dynindent,
  numwidth = 0pt,
  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`

Note. *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}` .

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

Note. *The environments presented in this section* [3] *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.*

1. Examples

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

1. The 1st argument between brackets `<...>` can take the values `nb` to number, which is the default setting, and `nonb` to not number.
2. The 2nd 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}

\begin{tdocexa}[Wonderful]
    Bli, bli, bli...
\end{tdocexa}

\begin{tdocexa}<nonb>
    Blo, blo, blo...
\end{tdocexa}

\begin{tdocexa}<nonb>[Superb]
    Blu, blu, blu...
\end{tdocexa}
```

Bla, bla, bla...

Example 1. *Ble, ble, ble...*

Example 2 (Wonderful)**.** *Bli, bli, bli...*

Example. *Blo, blo, blo...*

Example (Superb)**.** *Blu, blu, blu...*

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. Here's how to do it (this trick remains valid for the environments presented in the following sub-sections). Note in passing that the numbering follows that of the previous example as desired.*

[3] The formatting comes from the `amsthm` package.

```
\begin{tdocexa}
    \leavevmode

    \begin{enumerate}
        \item Point 1.

        \item Point 2.
    \end{enumerate}
\end{tdocexa}
```

Example 3.

1. *Point 1.*
2. *Point 2.*

2. Some remarks

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

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

\begin{tdocrem}[Mini title]
    Useful?
\end{tdocrem}
```

Remark. *Just one remark...*

Remark (Mini title)**.** *Useful?*

3. A tip

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

```
\begin{tdoctip}
    A tip.
\end{tdoctip}

\begin{tdoctip}[Mini title]
    Useful?
\end{tdoctip}
```

Tip. *A tip.*

Tip (Mini title)**.** *Useful?*

4. Informative note

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

```
\begin{tdocnote}
    Something useful to tell you...
\end{tdocnote}

\begin{tdocnote}[Mini title]
    Useful?
\end{tdocnote}
```

Note. *Something useful to tell you...*

Note (Mini title)**.** *Useful?*

5. Something important

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

```
\begin{tdocimportant}
    Important and harmless.
\end{tdocimportant}

\begin{tdocimportant}[Mini title]
    Useful?
\end{tdocimportant}
```

Important. *Important and harmless.*

Important (Mini title). *Useful?*

6. 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.

```
\begin{tdoccaution}
    Caution, caution...
\end{tdoccaution}

\begin{tdoccaution}[Mini title]
    Useful?
\end{tdoccaution}
```

Caution. *Caution, caution...*

Caution (Mini title). *Useful?*

7. 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.

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

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

Warning. *Avoid the dangers...*

Warning (Mini title). *Useful?*

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...

\tdocenv{env} produces...

We also have :

\tdocenv[{[opt1]<opt2>}]{env}
```

myclass is for...
mypackage is for...
\onemacro is for...
\begin{env} ... \end{env} produces...
We also have :
\begin{env}[opt1]<opt2> ...
\end{env}

Remark. *The advantage of the previous macros over the use of* `\tdocinlatex`*, see the section 1. page 11, is the absence of colouring. Furthermore, the* `\tdocenv` *macro simply asks you to type the name of the environment* [4] *with any options by typing the correct delimiters* [5] *by hand.*

Warning. *The optional argument to the* `\tdocenv` *macro is copied and pasted during rendering. This can sometimes require the use of protective braces, as in the previous example.*

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...

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

sup relates to...
sup·**erbe** means...
*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.

[4] In addition, `\tdocenv{monenv}` produces `\begin{monenv} ... \end{monenv}` with spaces to allow line breaks if necessary.

[5] Remember that almost anything is possible from now on.

1. With a coloured stripe

Example 1 (With default text)**.** *It can be useful to show a real rendering directly in a document* [6]*. 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 [7]*.*

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.*

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

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

```
\begin{tdocshowcase}[before = My beginning,
                     after  = My end,
                     color  = red]
Bla, bla, bla, bla, bla, bla, bla, bla, bla, bla, bla, bla, bla...
\end{tdocshowcase}
```

This will produce the following.

My beginning

Bla, bla, bla, bla, bla, bla, bla, bla, bla, bla, bla, bla, bla...

My end

Note. *You will have noticed that we don't obtain a pure red: behind the scenes, the expandable macros* `\tdocbackcolor` *and* `\tdocdarkcolor` *are used to create the background and title colours respectively from the ones proposed in* `\begin{tdocshowcase} ... \end{tdocshowcase}`*. These macros have a single argument, the chosen colour, and accept the following codes.*

[6] Typically when making a demo.

[7] Behind the scenes, the strip is created effortlessly using the `clrstrip` package.

```
\NewExpandableDocumentCommand{\tdocbackcolor}{m}{#1!5}
\NewExpandableDocumentCommand{\tdocdarkcolor}{m}{#1!50!black}
```

Warning. *With the default settings, if the code to be formatted begins with an opening bracket, an empty option must be explicitly indicated, as in the following example.*

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

This will produce the following.

Start of the real output

[This works...]

End of the real output

Note. *Behind the scenes, the* `\tdocruler` *macro is used.*

```
\tdocruler{Un pseudo-titre décoré}{red}
```

Un pseudo-titre décoré

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]
    Bla, bla, bla, bla, bla, bla, bla, bla, bla, bla, bla, bla, bla...
\end{tdocshowcase}
```

This will produce the following.

Start of the real output

Bla, bla, bla, bla, bla, bla, bla, bla, bla, bla, bla, bla, bla...

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]
Bla, bla, bla, bla, bla, bla, bla, bla, bla, bla, bla, bla, bla...
\end{tdocshowcase}
```

This will produce the following.

My beginning

Bla, bla, bla, bla, bla, bla, bla, bla, bla, bla, bla, bla, bla...

My end

3. 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. *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 `\tdocshowcaseinput[color = orange]{external.tex}` *, this will produce 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 a class is done efficiently using use cases showing both the code and the corresponding result.

1. "*Inline*" codes

The `\tdocinlatex` macro [8] 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|$a^b =
↪  c$|+
```

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

Note. *The* `\tdocinlatex` *macro can be used in a footnote: see the bottom of this page* [9].

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.*

[8] The name of the macro `\tdocinlatex` comes from "*tdocprewhyin.line LaTeX*".

[9] `$minted = TOP$` was typed `\tdocinlatex+$minted = TOP$+` in this footnote.

```
\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` [10].

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

$A = B + C$

Example 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.

```
[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, is used in the same way as the `\begin{tdoclatex} ... \end{tdoclatex}` environment except that the path to a file is supplied.

[10] `std` refers to the "*standard*" behaviour of `tcolorbox` in relation to the `minted` library.

Example 1 (Side by side).

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

This produces the following layout.

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

$xyz = 1$

Example 2 (Following).

```
\tdoclatexinput{examples-listing-xyz.tex}
```

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).

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

This produces the following layout.

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

4. Imported codes put into practice

Example 1 (Showcase). *The following can be obtained via* `\tdoclatexshow{examples-listing-xyz.t`

Start of the rendering in this doc.

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

This gives:

Start of the real output

$xyz = 1$

End of the real output

End of rendering in this doc.

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.
$x y z = 1$
```

Here is the actual rendering.

Start of the real output

$xyz = 1$

End of the real output

End of rendering in this doc.

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

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.*

```
Bla, bla, bla, bla, bla, bla, bla, bla, bla, bla, bla, bla, bla...

\medskip % CAUTION! This prevents overlapping.

\tdocdate{2023-09-24}

Ble, ble, ble, ble, ble, ble, ble, ble, ble, ble, ble, ble, ble...

\medskip % CAUTION! This prevents overlapping.

\tdocdate[gray]{2020-05-08}

Bli, bli, bli, bli, bli, bli, bli, bli, bli, bli, bli, bli, bli...

Blo, blo, blo, blo, blo, blo, blo, blo, blo, blo, blo, blo, blo...

Blu, blu, blu, blu, blu, blu, blu, blu, blu, blu, blu, blu, blu...
```

This gives:

Start of the real output

Bla, bla, bla, bla, bla, bla, bla, bla, bla, bla, bla, bla, bla...

2023-09-24 *Ble, ble, ble, ble, ble, ble, ble, ble, ble, ble, ble, ble, ble...*

2020-05-08 *Bli, bli, bli, bli, bli, bli, bli, bli, bli, bli, bli, bli, bli...*
Blo, blo, blo, blo, blo, blo, blo, blo, blo, blo, blo, blo, blo...
Blu, blu, blu, blu, blu, blu, blu, blu, blu, blu, blu, blu, blu...

End of the real output

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

```
\tdocversion[red]{10.2.0-beta}[2023-12-01]

Bla, bla, bla, bla, bla, bla, bla, bla, bla, bla, bla, bla, bla...

\bigskip % CAUTION! This prevents overlapping.

\tdocversion{10.2.0-alpha}

Ble, ble, ble, ble, ble, ble, ble, ble, ble, ble, ble, ble, ble,
ble, ble, ble, ble, ble, ble, ble, ble, ble, ble, ble, ble, ble,
ble, ble, ble, ble, ble, ble, ble, ble, ble, ble, ble, ble, ble,
```

```
ble, ble, ble, ble, ble, ble, ble, ble, ble, ble, ble, ble...
```

This gives:

Start of the real output

2023-12-01
10.2.0-beta

Bla, bla, bla, bla, bla, bla, bla, bla, bla, bla, bla, bla, bla...

10.2.0-alpha

Ble, ble, ble, ble, ble, ble, ble, ble, ble, ble, ble, ble, ble, ble, ble, ble, ble, ble, ble, ble, ble, ble, ble, ble, ble, ble, ble, ble, ble, ble, ble, ble, ble, ble, ble, ble, ble, ble, ble, ble, ble, ble, ble, ble, ble, ble, ble, ble, ble...

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.* [11], *and this is a choice! Why? Quite simply because dating and versioning explanations should be done semi-automatically to avoid any human bugs.*

2. What's new?

`tutodoc` offers different environments to indicate quickly and clearly what has been done [12] during the latest changes.

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}
    \item Info 1...
    \item Info 2...
\end{tdocupdate}
```

Update.

- *Info 1...*
- *Info 2...*

Example 3 (For fixes)**.**

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

Fix.

- *Info 1...*
- *Info 2...*

[11] Technically, checking the validity of a date using LaTeX3 presents no difficulty.
[12] The user doesn't need all the technical details.

Example 4 (Chosen topics)**.**

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

Unclassifiable changes.

- *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.

Ble, ble, ble...

Bli, bli, bli...

\tdocxspace % Subtle space
            % but useful.

Blo, blo, blo...

Blu, blu, blu...
```

Bla, bla, bla...

Ble, ble, ble...
Bli, bli, bli...

Blo, blo, blo...
Blu, blu, blu...

XI. History

2023-11-29
1.0.0

First public version of the project.