The `tutodoc` class
Tutorial-style documentation

Christophe, BAL

Dec 18, 2024 - Version 1.7.1

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

Remark : *this documentation is also available in French.*

Last changes

Fix.

- Documentation: references to tools to indicate changes have been incorrectly written as characteristics of highlighted colored content.

Break.

- The `\tdocenv` macro and its starred version no longer offer an option.
- LaTeX showcases: the default layout is more sober, and there are options for having just the rulers, or the colored stripe. See just after.

New.

- Formatting of computer codes in addition to those specifically in LaTeX.
  1. Creation of `\begin{tdoccode}...\end{tdoccode}` and `\tdoccodein`.
  2. For macros for inline code, and environments for blocks of code, `minted` options are indicated inside square brackets in the traditional way: `[minted options]`.
  3. For code block environments, `tcolorbox` options are indicated inside rafters: `<tcolorbox options>`.
  4. The new macro `\tdoctcb` allows to use shortcuts for regularly used `tcolorbox` styles.
- Documentation: a new section presents tools for formatting computer codes other than those in LaTeX.

Update.

- Sub-sub-sections are numbered in lower case.
- Themes.
  1. Less space consumed.
  2. Shadows have better coloring.
  3. For all themes except the `draft` one, the radius of the arcs of the corners of the frames has changed from `.75mm` to `.2pt`,.
  4. Use case in LaTeX: with the theme `color`, the background color changes from `yellow!4` to `gray!5`.
  5. Latest changes: with the `dark` theme, the `[Init]` text produced by the `\tdocstartproj` macro uses the same font as the environment titles to indicate changes.

[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 for a coding documentation.

Contents

I. Dependencies **4**

II. General settings **4**
1. Font size and page geometry 4
2. Titles and table of contents 4
3. Dynamic links 4

III. What language is used by the `tutodoc` class? **4**

IV. What does that mean in "*English*"? **5**

V. Choose your theme **5**

VI. Highlighting content **5**
1. Content in the reading flow 5
   a. Examples 5
   b. Some remarks 6
2. Flashy content 6
   a. A tip 6
   b. Informative note 7
   c. Something important 7
   d. Caution about a delicate point 7
   e. Warning of danger 7

VII. Specify packages, classes, macros or environments **8**

VIII. Origin of a prefix or suffix **8**

IX. A real-life rendering **8**
1. A minimalist rendering by default 8
2. With framing lines 9
3. With colored stripe 9
4. By importing the LaTeX code 10

X. 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

XI. Presenting computer code **14**
1. "*Inline*" codes 14
2. Codes typed directly 15
3. Imported codes 16

XII. Indicate changes **16**
1. When? 16
2. What's new? 18
   a. Sobriety first 18
   b. Color if necessary 19
3. The what and the when 20

XIII. Ornament **20**

XIV. Contribute **21**
1. Complete the translations 21
   a. The `fr` and `en` folders 21
   b. The `changes` folder 21
   c. The `status` folder 21
   d. The `README.md` and `LICENSE.txt` files 21
   e. New translations 22
2. Improving the source code 22

XV. History 23

Appendix – Theme gallery 26

I. Dependencies

`tutodoc` admits the following dependencies (the dates in brackets are those of the versions used during the latest tests).

- `scrartcl.cls` (2024/10/24)
- `csquotes.sty` (2024/04/04)
- `geometry.sty` (2020/01/02)
- `inputenc.sty` (2024/02/08)
- `marginnote.sty` (2018/08/09)
- `tcolorbox.sty` (2024/10/22)
- `clrstrip.sty` (2021/08/28)
- `fontawesome5.sty` (2022/05/02)
- `hyperref.sty` (2024/11/05)
- `keytheorems.sty` (2024/11/11)
- `minted.sty` (2024/11/17)

II. General settings

1. Font size and page geometry

The `scrartcl` class is loaded via the `fontsize = 10pt` option, and the `geometry` package manages the page dimensions.

Warning.

The macros for dating and versioning presented in the section XII on page 16 require fixed settings for page geometry and font size.

2. Titles and table of contents

The selected settings are directly visible in this documentation.

3. Dynamic links

The `hyperref` package is imported, if it hasn't already been, and the settings chosen are just for the colors of links relating to citations, files, internal links, and finally `url` (this colors will depend on the theme chosen).

III. What language is used by the `tutodoc` class?

This documentation loads the `babel` package via `\usepackage[english]{babel}` a package that `tutodoc` does not load. On the other hand, 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.

- `en` : English.
- `es` : Spanish.
- `fr` : French.

Note.

Packages `babel` and `polyglossia` are taken into account.

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`.

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

IV. 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 detected by `tutodoc`*, the macro* `\tdocinEN` *and its starred version become useful for non-English speakers.*

V. Choose your theme

To modify the general layout, there is the `tutodoc` class option `theme = <choice>` where `<choice>` can take the following values.

- `bw`: a black-and-white theme with some shades of grey.
- `color`: a colored theme, this is the default value.
- `dark`: a dark theme ideal for resting the eyes.
- `draft`: a theme for a printout such as to look for content errors that aren't necessarily easy to spot in front of a screen.

Note.

At the end of this document, after the change history, you'll find a gallery of use cases for these different themes : go to appendix page 26.

VI. Highlighting content

Note.

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, which will be reset to zero as soon `\section` *is used.*

a. Examples

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

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

\begin{tdocexa}[Small title]
    Useful?
\end{tdocexa}
```

Example VI.1. *An example...*

Example VI.2 (Small title)**.** *Useful?*

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 VI.3.

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

b. 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}

\begin{tdocrem}
    Another?
\end{tdocrem}

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

Remark VI.4. *Just one remark...*

Remark VI.5. *Another?*

Remark VI.6 (Small title)**.** *Useful?*

2. Flashy content

Note.

The formatting proposed here is the default one, but others are possible by changing the theme: see the gallery of use cases in the appendix page 26. As for the icons, they are obtained via the `fontawesome5` *package, and the* `\tdocicon` *macro which manages the spacing relatively to the text.*[a]

[a] For example, `\fbox{tdocicon{faBed}{Fatigued}}` produces Fatigued.

a. A tip

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

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

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

Tip.

A tip.

Tip (Small title)**.**

Useful?

Tip.

Sometimes, highlighted content can be reduced to a list. In this case, the formatting can be improved as follows where we use the `wide` *option from the* `enumitem` *package imported by this documentation.*

```latex
\begin{tdoctip}[Little elegant]
    \begin{enumerate}
        \item Point 1.
        \item Point 2.
    \end{enumerate}
\end{tdoctip}
VERSUS.
\begin{tdoctip}[More elegant]
    \begin{enumerate}[wide]
        \item Point 1.
        \item Point 2.
    \end{enumerate}
\end{tdoctip}
```

Tip (Little elegant).

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

VERSUS.

Tip (More elegant).

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

b. Informative note

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

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

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

Note.

Something useful to tell you...

Note (Small title).

Useful?

c. Something important

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

```latex
\begin{tdocimp}
    Important and harmless.
\end{tdocimp}

\begin{tdocimp}[Small title]
    Useful?
\end{tdocimp}
```

Important.

Important and harmless.

Important (Small title).

Useful?

d. Caution about a delicate point

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

```latex
\begin{tdoccaut}
    Caution, caution...
\end{tdoccaut}

\begin{tdoccaut}[Small title]
    Useful?
\end{tdoccaut}
```

Caution.

Caution, caution...

Caution (Small title).

Useful?

e. 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}[Small title]
    Useful?
\end{tdocwarn}
```

Warning.

Avoid the dangers...

Warning (Small title).

Useful?

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

myclass is for...
mypackage is for...
\onemacro is for...
\begin{env} ...\end{env} produces...
Just env...

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

VIII. 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 VIII.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`*.*

IX. A real-life rendering

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

1. A minimalist rendering by default

Example IX.1. *It can be useful to show a real rendering directly in a document.*[4] *This is typed via the environment* `tdocshowcase` *as follows.*

```
\begin{tdocshowcase}
    \bfseries A bit of code \LaTeX.

    \bigskip

    \emph{\large End of the awful demo.}
\end{tdocshowcase}
```

This results in the following rendering, which is a combination of low vertical spacing and simple import.

A bit of code LaTeX.

End of the awful demo.

Remark IX.2. *The section 4 on page 13 explains how to obtain, via the macro* `\tdoclatexshow`*, a code followed by its actual rendering as in the previous example.*

[4] Typically when making a demo.

Warning.

With the default settings, if the code to be formatted begins with an opening bracket, we must use one of the following tricks.

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

OR.

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

This will produce the following.

[This works...]
OR.
[This also works...]

2. With framing lines

To make the formatted LaTeX code more visible, you can use the `rule` style, as in the following examples.

Example IX.3. *The option* `style = rule` *provides the following where the automatically added texts will adapt to the language found by* `tutodoc`.

Start of the real output

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

End of the real output

Example IX.4 (Editable text and colours)**.** *You can easily obtain the following horror.*

My beginning

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

My end

Here's the code that was used.[5]

```
\begin{tdocshowcase}[style      = rule,
                     col-stripe = red,
                     col-text   = orange!75!black,
                     before     = My beginning,
                     after      = My end]
    Bla, bla, bla, bla, bla, bla, bla, bla, bla, bla, bla, bla, bla...
\end{tdocshowcase}
```

Note.

In the previous example, the text uses the proposed darkened orange. However, the red is used as a base to obtain the colors used for the framing lines: the transformations used depend on the theme chosen.[a] *You should also be aware that behind the scenes, the macro* `\tdocruler` *is used, it works as follows.*

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

A decorated pseudo-title

[a] For example, the themes `bw` and `draft` ignore the key `col-stripe`!

3. With colored stripe

There are situations where you need to be able to clearly identify an example of formatted LaTeX code. This can be done, as the following examples show.[6]

[5] The next section will justify the a priori strange choice of `col-stripe` instead of `col-rule`.
[6] Behind the scenes, the strips are created effortlessly using the `clrstrip` package.

Example IX.5. *The* `style = stripe` *option provides 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 IX.6 (Editable text and colors)**.** *You can easily produce a beautiful horror like the one below.*

Mon début

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

Ma fin à moi

Here's the code that was used.[7]

```
\begin{tdocshowcase}[style      = stripe,
                     col-stripe = green,
                     col-text   = purple,
                     before     = Mon début,
                     after      = Ma fin à moi]
    Bla, bla, bla, bla, bla, bla, bla, bla, bla, bla, bla, bla, bla...
\end{tdocshowcase}
```

4. By importing the LaTeX code

To obtain renderings by importing the code from an external file, instead of typing it, simply use the macro `\tdocshowcaseinput` whose option uses the same syntax as that of the environment `tdocshowcase`, and the mandatory argument corresponds to the path of the file. Here are some examples of use.

Example IX.7 (Standard use)**.**

```
\tdocshowcaseinput{examples-showcase-external.tex}
```

This gives:

Blablobli, blablobli, blablobli, blablobli, blablobli, blablobli...

Example IX.8 (With framing lines)**.**

```
\tdocshowcaseinput[style = rule]{examples-showcase-external.tex}
```

This gives:

Start of the real output

Blablobli, blablobli, blablobli, blablobli, blablobli, blablobli...

End of the real output

Example IX.9 (A colored stripe)**.**

```
\tdocshowcaseinput[style      = stripe,
                   col-stripe = red,
                   col-text   = LightCoral]%
                  {examples-showcase-external.tex}
```

This gives:

Start of the real output

Blablobli, blablobli, blablobli, blablobli, blablobli, blablobli...

End of the real output

X. Use cases in LaTeX

Documenting a package, or class, is best done through use cases showing both the code and the corresponding result.[8]

[7] Now we understand why we chose `col-stripe` instead of `col-rule`.
[8] Code is formatted using the `minted` and `tcolorbox` packages.

1. "*Inline*" codes

Example X.1 (Standard use). *The* `\tdoclatexin` *macro*[9] *can be used to type code in line in a similar way to* `\verb`*, or as a standard macro (see the handling of braces in the latter case below). Here are some examples of use.*[10]

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

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

Example X.2 (Possible options). *As the* `\tdoclatexin` *macro is based on* `minted`*, you can use all the options taken into account by* `minted`*. Here are some examples.*

```
1: \tdoclatexin[style = bw]{$a^b = c$} \\
2: \tdoclatexin[style = igor,
                showspaces]{$a^b = c$}
```

```
1: $a^b = c$
2: $a^b␣=␣c$
```

Note.

The `\tdoclatexin` *macro can be used in a footnote as shown below.*[a]

[a] `$minted = TOP$` has been typed `\tdoclatexin+$minted = TOP$+` in this footnote.

2. Directly typed codes

Example X.3 (Side by side). *Displaying a code and its rendering side by side is done as follows where the macro* `\tdoctcb` *allows you to just type* `tdoctcb{sbs}` *instead of* `listing side text` *(*`sbs` *is for "*`s·ide b·y s·ide`*", while* `tcb` *is the standard abbreviation for* `tcolorbox`*). Note the use of rafters, not square brackets (more on this later).*

```
\begin{tdoclatex}<\tdoctcb{sbs}>
$A = B + C$
\end{tdoclatex}
```

This gives :

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

$A = B + C$

Example X.4 (Following). `\begin{tdoclatex}...\end{tdoclatex}` *produces the following result (this default setting is also obtained by using* `\tdoctcb{std}`*).*[11]

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

$A = B + C$

Example X.5 (Just the code). *Via* `\tdoctcb{code}`*, we'll just get the code as below.*

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

Example X.6 (Customise). *The* `tdoclatex` *environment accepts two types of optional argument.*

1. *Between classic square brackets, you can use any option taken into account by* `minted`*.*
2. *Between rafters, you can use any option managed by the environments obtained via* `tcolorbox`*.*

For example, the following modifications can be made if required.[12]

```
\begin{tdoclatex}%
    [linenos, style = igor, showspaces]%
    <\tdoctcb{sbs},
     attach boxed title to top left = {yshift = -9pt},
     fonttitle                      = \bfseries,
     title                          = Local modifications,
```

[9] The name of the macro `\tdoclatexin` comes from "*in·line* LaTeX".
[10] A background color is deliberately used to subtly highlight the `\LaTeX` codes.
[11] `std` refers to the "*standard*" behaviour of `tcolorbox` in relation to the `minted` library.
[12] This documentation uses the options between rafters to obtain correct rendering of code producing shaded frames: see the section 2 on page 6.

```
    top                                = 10pt>
% Sometimes useful, but don't overuse it!
$A = B + C$
% End of this demonstration.
\end{tdoclatex}
```

This gives :

Local modifications

```
% Sometimes useful, but don't overuse it!
$A = B + C$
% End of this demonstration.
```

$A = B + C$

Warning.

To obtain the default formatting for a code beginning with a bracket or a rafter, you'll need to do a bit of fiddling, as shown below.

```
\begin{tdoclatex}[]
[Strange... Or not!]
\end{tdoclatex}
OR.
\begin{tdoclatex}<>
<Strange... Or not!>
\end{tdoclatex}
```

This gives :

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

[Strange... Or not!]

OR.

```
<Strange... Or not!>
```

<Strange... Or not!>

Another method is to use the `\string` *primitive, as shown below.*

```
\begin{tdoclatex}
\string[Strange... Or not!]
\end{tdoclatex}
OR.
\begin{tdoclatex}
\string<Strange... Or not!>
\end{tdoclatex}
```

This gives :

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

[Strange... Or not!]

OR.

```
<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 system of options between square brackets, or rafters, as the environment tdoclatex.

Example X.7 (Side by side).

```
\tdoclatexinput<\tdoctcb{sbs}>{examples-listing-latex-xyz.tex}
```

This produces the following formatting.

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

$xyz = 1$

Example X.8 (Following).

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

This produces the following formatting, which also corresponds to the option \tdoctcb{std}.

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

$xyz = 1$

Example X.9 (Only the code).

```
\tdoclatexinput<\tdoctcb{code}>{examples-listing-latex-xyz.tex}
```

This produces the following formatting.

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

Example X.10 (Customise).

```
\tdoclatexinput[style = igor, showspaces]%
                <colframe = purple, colback = red!5>%
                {examples-listing-latex-xyz.tex}
```

This produces the following formatting.

```
%␣Just␣one␣demo.
$x␣y␣z␣=␣1$
```

$xyz = 1$

4. Imported codes put into practice

Note.

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

Example X.11 (Showcase). *The following comes from* \tdoclatexshow{examples-listing-xyz.tex}.

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

This gives :

$xyz = 1$

Example X.12 (Changing the explanatory text). *Using the key* explain, *you can use a custom text. Thus,* \tdoclatexshow[explain = Here is the rendering.]{examples-listing-xyz.tex} *will give the following.*

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

Here is the rendering.

$xyz = 1$

Example X.13 (The options available)**.** *In addition to the explanatory text, it is also possible to use all the options of* `tdocshowcase` *environment, see IX on page 8. Here is an example to illustrate this.*

```
\tdoclatexshow[style      = stripe,
               col-stripe = orange,
               col-text   = blue!70!black,
               before     = Rendering hereafter.,
               explain    = What comes next is coloured...,
               after      = Finished rendering.,]
               {examples-listing-latex-xyz.tex}
```

This will produce the following.

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

What comes next is coloured...

Rendering hereafter.

$xyz = 1$

Finished rendering.

XI. Presenting computer code

Some packages offer functions that require to code a little in `Lua`.[13] For these projects, the documentation must be able to present lines of code; this is why `tutodoc` makes it easy to do this, and much more.[14]

Important.

The tools in this section can also be used to present LaTeXcode, but they should not be used for simple use cases, as the macros and environments presented next are for studying code, not just for using it: see the section X on page 10 to use the right tools for formatting LaTeX use cases.

1. "*Inline*" codes

The `\tdoccodein`[15] macro expects two arguments: the 1st indicates the programming language, and the 2nd gives the code to be formatted. It is possible to use an option identical to that proposed by `\tdoclatexin`: see the section 1 on page 11. Here are some possible use cases.[16]

```
1: \tdoccodein{py}{print("OK" if i = 0 else "KO")}             \\
2: \tdoccodein[style = bw]{py}{print("OK" if i = 0 else "KO")} \\
3: \tdoccodein[style = igor, showspaces]%
              {py}{print("OK" if i = 0 else "KO")}
```

```
1: print("OK" if i = 0 else "KO")
2: print("OK" if i = 0 else "KO")
3: print("OK"␣if␣i␣=␣0␣else␣"KO")
```

[13] For mathematics, these include `luacas` and `tkz-elements`.

[14] As code formatting is done via the packages `minted` and `tcolorbox`, the macros and environments presented in this section allow code to be formatted in all the languages supported by `Pygments`, a `Python` project used behind the scenes by `minted`.

[15] The name of the macro `\tdoccodein` comes from "*in·line code*".

[16] A background color is used to subtly highlight the formatted codes. For example, typing `\tdoccodein{py}{funny = "ah"*3}` will produce `funny = "ah"*3`.

Note.

The page `https://pygments.org/languages` *contains a complete list of supported languages with their short names. For example, it is possible to format* `Brainfuck` *code like this obscure sequence* `++++++++++[>+++++++>++++++++++>+++>+<<<<-]>++.>+.+++++++..+++.` *which displays* `Hello`*.*

2. Codes typed directly

Code can be typed directly into a document via `\begin{tdoccode}...\end{tdoccode}` which expects an argument indicating the programming language, and any options between parenthesis and/or square brackets identical to those proposed by `\begin{tdoclatex}...\end{tdoclatex}`: see the section X on page 10.[17]

Example XI.1 (Standard feature)**.**

```
\begin{tdoccode}{pl}
print "Who are you? ";
my $name = <STDIN>;

chomp($name);

if ($name eq "") {
    print "Ah, not very chatty today!";

} else {
    print "Hello $name";
    print "Amazing! Actually, not at all...";
}
\end{tdoccode}
```

This gives :

```
print "Who are you? ";
my $name = <STDIN>;

chomp($name);

if ($name eq "") {
    print "Ah, not very chatty today!";

} else {
    print "Hello $name";
    print "Amazing! Actually, not at all...";
}
```

Example XI.2 (One-off rendering customization)**.**

```
\begin{tdoccode}[style = solarized-light, linenos]%
                <leftrule = 22pt, colback = orange!5, colframe = red!35>%
                {lua}
io.write("Who are you?")
local name = io.read()

if name == "" then
    print("Ah, not very chatty today!")

else
    print("Hello “ .. name .. ”.")
    print("Amazing! Actually, not at all...")
end
\end{tdoccode}
```

This gives :

[17] Note that the coloring of the LaTeX codes is lexically correct, but semantically wrong.

```lua
io.write("Who are you?")
local name = io.read()

if name == "" then
    print("Ah, not very chatty today!")

else
    print("Hello “ .. name .. ”.")
    print("Amazing! Actually, not at all...")
end
```

3. Imported codes

The `tdoccodeinput` macro expects the language and path of a file to be formatted, and possibly options similar to those offered by the `tdoccode` environment.

Example XI.3 (Standard features)**.**

```
\tdoccodeinput{hs}{examples-listing-full-hello-you.hs}
```

This gives:

```haskell
main :: IO ()

main = do
    putStr "Who are you? "
    name <- getLine

    if name == ""
        then putStrLn "Ah, not very chatty today!"

    else do
        putStrLn ("Hello " ++ name ++ ".")
        putStrLn "Amazing! Actually, not at all..."
```

Example XI.4 (Customize rendering on occasion)**.**

```
\tdoccodeinput[style = solarized-light, linenos]%
              <leftrule = 22pt, colback = orange!5, colframe = red!35>%
              {tex}{examples-listing-full-hello-you.tex}
```

This gives:

```tex
\NewDocumentCommand{\helloyou}{m}{%
    \IfBlankTF{#1}{%
        Ah, not very chatty today!
    }{%
        Hello #1.

        Amazing! Actually, not at all...%
    }%
}
```

XII. 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 date and/or version something.

Example XII.1 (Dating new features)**.** *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 :

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...

Example XII.2 (Versioning new features, possibly with a date)**.** *Associating a version number with a new feature is done using the* `\tdocversion` *macro, with the color 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...

\smallskip\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, ble...
```

This gives :

10.2.0-beta 2023-12-01 *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, ble, ble...*

Example XII.3 (Caution with paragraph titles)**.** *The following example shows that a date and/or version must be placed just after a paragraph title, and not before it.*

```
\paragraph{A well-versioned title.}
\tdocversion{1.2.3}[2024-11-23]
Blah, blah, blah, blah, blah, blah, blah, blah, blah, blah, blah...
Stay, stay, stay, stay, stay, stay, stay, stay, stay, stay, stay, stay, stay, stay...

Stay, stay, stay, stay, stay, stay, stay, stay, stay, stay, stay, stay, stay...
Blo, blo, blo, blo, blo, blo, blo, blo, blo, blo, blo, blo, blo, blo...

\tdocdate{2024-11-23}
\paragraph{A badly versioned title.}
Blu, blu, blu, blu, blu, blu, blu, blu, blu, blu, blu, blu, blu, blu...
```

This gives :

1.2.3
2024-11-23

A well-versioned title. *Blah, blah, blah, blah, blah, blah, blah, blah, blah, blah, blah... Stay, stay, stay, stay, stay, stay, stay, stay, stay, stay, stay, stay, stay, stay, stay...*
Stay, stay, stay, stay, stay, stay, stay, stay, stay, stay, stay, stay, stay... Blo, blo, blo, blo, blo, blo, blo, blo, blo, blo, blo, blo, blo, blo...

2024-11-23

A badly versioned title. *Blu, blu, blu, blu, blu, blu, blu, blu, blu, blu, blu, blu, blu, blu...*

Example XII.4 (Adjust vertical shift)**.** *If required, you can modify the vertical offset used to place dates and versions in the margin, the default value being* $(-8\,pt)$.

```
This is what it looks like without vertical shift.

\paragraph{A home-made setting.}%
\tdocversion{1.2.3}[2024-10-29]<0pt>

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

This gives :

This is what it looks like without vertical shift.

1.2.3
2024-10-29

A home-made setting. *Blo, blo, blo, blo, blo, blo, blo, blo, blo, blo, blo, blo...*

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`*.*

Caution.

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.*

[a] Technically, checking the validity of a date using LaTeX3 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 changes made, or to come.[18]

Note.

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

a. Sobriety first

Example XII.5 (Just for the very first version)**.**

```
\tdocstartproj{1st version of the project.}
```

1st version of the project.

Example XII.6 (For new features)**.**

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

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

Example XII.7 (For updates)**.**

[18] The user doesn't need all the technical details.

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

Update.

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

Example XII.8 (For breaks)**.**

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

Break.

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

Example XII.9 (For problems)**.**

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

Problem.

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

Example XII.10 (For fixes)**.**

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

Fix.

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

Example XII.11 (Roadmap)**.**

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

Todo.

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

Example XII.12 (Technical information)**.**

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

Technical information.

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

Example XII.13 (Selectable themes with an icon)**.**

```
\begin{tdoctopic}{To hide}<\faEyeSlash>
% An icon from fontawesome5.
    \item Info 1...
    \item Info 2...
\end{tdoctopic}
```

To hide.

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

Example XII.14 (Selectable themes without icons)**.**

```
\begin{tdoctopic}{End of icons}
    \item Info 1...
    \item Info 2...
\end{tdoctopic}
```

End of icons.

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

b. Color if necessary

It may be useful to highlight some changes: this can only be done by modifying the content color.

Example XII.15 (A flashy first version)**.**

```
\tdocstartproj[DarkOrchid]%
              {Brightly colored version 1.}
```

Brightly colored version 1.

Example XII.16 (Outstanding fixes)**.**

```
\begin{tdocfix}[col = CadetBlue]
    \item Info...
\end{tdocfix}
```

Fix.
- *Info...*

3. The what and the when

The optional keys `col-chges`, `date` and `version` allow to date and/or version a change of a particular type. Here are some examples of use.

```
\begin{tdoctech}[date      = 2024-10-29,
                 col-chges = red]
    \item Info...
\end{tdoctech}

\begin{tdocupdate}[version   = 1.2.3,
                   col-chges = ForestGreen,
                   col       = ForestGreen]
    \item Info...
\end{tdocupdate}

\begin{tdoctopic}{To hide}<\faEyeSlash>%
                 [version = 4.5.6,
                  date    = 2025-11-30]
    \item Info...
\end{tdoctopic}
```

This gives :

2024-10-29 **Technical information.**

- Info...

1.2.3 **Update.**

- Info...

4.5.6 2025-11-30 **To hide.**

- Info...

XIII. Ornament

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

```
Bla, bla, bla...

\tdocsep % Practical for demarcation.

This works with enumerations.

\begin{itemize}
    \item Focus.
\end{itemize}

\tdocsep % Uniform behaviour.

Ble, ble, ble...
```

Bla, bla, bla...

This works with enumerations.

- Focus.

Ble, ble, ble...

XIV. Contribute

Note.

***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

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]

[a] The existence of a French version is simply a consequence of the native language of the author of `tutodoc`.

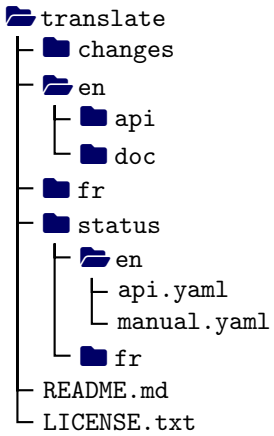

Figure 1: Simplified view of the translation folder

The translations are roughly organized as in figure 1 where just the important folders for the translations have been "*opened*".[19] **A little further down, the section e explains how to add new translations.**

a. 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.

b. 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.

c. 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.

d. 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.

[19] This was the organization on October 5, 2024.

e. New translations

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 document processor. The content of these files uses commented lines in English to explain what* `tutodoc` *will do; these lines begin with* `//` *. Here's an extract from such a file, where translations are made after each = sign, without touching the preceding, as this initial piece is used internally by the* `tutodoc` *code.*

```
// #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
```

Note.

The `doc` *folder is reserved for documentation. It contains* `TEX` *files 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.[20]

Method 1 : use of `git`.

1. Recover 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 an `it` copy of the `en` folder, with the short name of the language documented in the page "*IIETF language tag*" from `Wikipedia`.
3. Once the translation is complete in the `it` folder, share it via `https://github.com/bc-tools/for-latex/tree/tutodoc` using a classic `git push` .

Method 2 : communicate by e-mail.

1. 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

Important.

If you want to participate to `tutodoc`*, you'll need to use the LaTeX3 programming paradigm.*

[20] As mentioned above, there is no real need for the `doc` folder.

Participation as a coder is made via the repository https://github.com/bc-tools/for-latex/tree/tutodoc corresponding to the `tutodoc` development branch.

Caution.

Do not use the `main` *branch, which is used to freeze the latest stable versions of projects in the mono repository* https://github.com/bc-tools/for-latex.

XV. History

1.7.1
2024-12-18

Fix.

- Documentation: references to tools to indicate changes have been incorrectly written as characteristics of highlighted colored content.

Break.

- The `\tdocenv` macro and its starred version no longer offer an option.
- LaTeX showcases: the default layout is more sober, and there are options for having just the rulers, or the colored stripe. See just after.

New.

- Formatting of computer codes in addition to those specifically in LaTeX.
  1. Creation of `\begin{tdoccode}...\end{tdoccode}` and `\tdoccodein`.
  2. For macros for inline code, and environments for blocks of code, `minted` options are indicated inside square brackets in the traditional way: `[minted options]`.
  3. For code block environments, `tcolorbox` options are indicated inside rafters: `<tcolorbox options>`.
  4. The new macro `\tdoctcb` allows to use shortcuts for regularly used `tcolorbox` styles.
- Documentation: a new section presents tools for formatting computer codes other than those in LaTeX.

Update.

- Sub-sub-sections are numbered in lower case.
- Themes.
  1. Less space consumed.
  2. Shadows have better coloring.
  3. For all themes except the `draft` one, the radius of the arcs of the corners of the frames has changed from `.75mm` to `.2pt`,.
  4. Use case in LaTeX: with the theme `color`, the background color changes from `yellow!4` to `gray!5`.
  5. Latest changes: with the `dark` theme, the `[Init]` text produced by the `\tdocstartproj` macro uses the same font as the environment titles to indicate changes.

1.7.0
2024-12-04

Break.

- Format: the `scrartcl` class replaces the venerable `article`. This implies better placement of the margin notes with the options retained for loading `scrartcl`.
- LaTeX code: the macro `\tdocinlatex` has been renamed `\tdoclatexin`.
- Color key names will be hyphenated where necessary: this implies the following changes.
  1. Indicate the latest changes: the `colchges` option of the environments has been renamed `col-chges`.
  2. Showcases: for the environment `tdocshowcase` and the macro `\tdocshowcaseinput`, the `colstripe` and `coltext` options have been renamed `col-stripe` and `col-text`.

Fix.

- Admonitions: for the `\newkeytheorem` used with the `draft` theme, `postheadhook = \leavevmode` has been added (this is necessary because the content can naturally be of the list type).

New.

- Documentation: addition of a section listing dependencies.
- Class options.
  1. Options not specific to `tutodoc` are passed on to the class in charge of general formatting.
  2. The `scrartcl` options `fontsize` and `DIV` can't be used because their values are fixed by `tutodoc`.
- The macro `\tdocinEN` respects the English linguistic rules.
- Indicate the latest changes.
  1. Add the environment `\begin{tdoctodo}...\end{tdoctodo}`.
  2. Each environment has a new option `col` for the color of the content indicating changes.

Update.

- `draft` theme and changes: the environments for the latest changes stop to use icons.
- Documentation: the theme gallery uses a better fake example.

Technical information.

- Simplified organisation of configuration files in the final project.
  1. Use of one file per theme with a name like `tutodoc-*.css.cls`.
  2. Locale: use of names like `tutodoc-*.loc.cls`.

1.6.2
2024-10-30

New.

- The macros `\tdocdate` and `\tdocversion` has a new final optional argument `<voffset>` to choose a specific vertical offset.
- Better environments to indicate the changes made.
  1. The new optional keys `col`, `date` and `version` allow to date and version a change of a specific topic.
  2. Use of `\paragraph` for the title.

Update.

- Version and changes: the font of the margin notes will always have a normal shape.
- Ornament: use of a `\cleaders` to avoid orphean rules at the bottom of a page.

1.6.1
2024-10-28

Technical information.

- The naming rules of CTAN need the use of CSS files named `tutodoc-*.css.cls.sty`.

1.6.0
2024-10-27

Break.

- The `showcase` environment and its descendants: the `color` key has been renamed `colstripe`.
- The macro `\tdoclinkcolor` becomes the color `tutodoc@link@color` for internal use.

New.

- The `theme` class option allows you to choose different formatting themes.
- Change log: addition of the `tdoctech` environment for technical information.
- The `showcase` environment and its descendants: the `coltext` key can also be used to change the text color.
- The new functionalities have been documented.

Update.

- Change log: the `tdocupdate` environment uses the icon [icon] instead of [icon].

Fix.

- The Spanish translations were not included in the previous version! Don't laugh too loud...

1.5.0
2024-10-19

Technical information.

- Version 3 of `minted` is taken into account.

Break.

- The `tutodoc` class replaces the now-defunct `tutodoc` package (for the moment, the young class offers no specific options).
- The `\tdocruler` macro is now used via `\tdocruler[<color>]{<text>}` (remember that the old syntax was `\tdocruler{<text>}{<color>}`).

New.

- The class is usable in Spanish.
- The documentation contains a new section explaining how to contribute.

Fix.

- The `\tdocdate` macro did not handle date format and formatting.
- Colored frames did not color text after a page break.

1.4.0
2024-09-28

Break.

- The `tdoccaution` environment has been renamed `tdoccaut` for simplified input.
- Content highlighting: examples and remarks, indicated via the `tdocexa` and `tdocrem` environments, are numbered using 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.

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

Technical information.

- 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 temporarily 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`
  2. `tdocimp`
  3. `tdocnote`
  4. `tdoctip`
  5. `tdocwarn`

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}` ... `\end{tdocprob}` for identified problems.
- `\tdoclatexin`: 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{[}` ... `\end{[}`[nostripe]]tdocshowcase : page breaks around "*framing*" lines should be rare from now on.

1.0.0
2023-11-29

First public version of the project.

Appendix – Theme gallery

Note.

Each example is a `PDF` *directly inserted into this document (so don't be surprised by the page numbers).*

The "bw" theme

I. Liens

A very big link, but at least we can see it.

II. LaTeX listings

Typing inline code such as `E = m c^2 \neq \pi \neq \frac{3}{14}` is useful, as is showing use cases such as the following one.

```
Formatted \LaTeX\ code is great: $E = m c^2$ or $pi \neq \frac{3}{14}$.
```

Formatted LaTeX code is great: $E = mc^2$ or $pi \neq \frac{3}{14}$.

There's also a less invasive side-by-side mode. Nice! No ?

```
Formatted \LaTeX\ code is great:        \\
$E = m c^2$ or $\pi \neq \frac{3}{14}$.
```

Formatted LaTeX code is great:
$E = mc^2$ or $\pi \neq \frac{3}{14}$.

III. Highlighting, versioning and dating

1. tdocexa, tdocrem

1.7.0
2024-12-04

In the flow of the text, it is always useful to be able to provide examples and comments to supplement the main content.

Example III.1. *What to say*[1]*? I don't know, but it's nice. No ?*

Remark III.2. *What to say*[2]*? I don't know, but it's nice. No ?*

2. tdocnote, tdoctip...

Depending on the context of use, it is sometimes necessary to be able to highlight content by indicating its degree of importance.

Note.

What to say[a]*? I don't know, but it's nice. No ?*

[a] Let's not forget the footnotes...

Tip.

What to say? I don't know, but it's nice. No ?

Important.

What to say? I don't know, but it's nice. No ?

Caution.

What to say? I don't know, but it's nice. No ?

Warning.

What to say? I don't know, but it's nice. No ?

[1] Let's not forget the footnotes...
[2] Let's not forget the footnotes...

3. tdocbreak, tdocfix...

A new demonstration section...

Todo.

- A gallery would be welcome...

In a change log, it is important to visualise the types of changes clearly. This makes it easier for the user to read!

Break.

- Infos...

Fix.

- Infos...

New.

- Infos...

Problem.

- Infos...

Technical information.

- Infos...

Update.

- Infos...

Todo.

- Infos...

The "color" theme

I. Liens

A very big link, but at least we can see it.

II. LaTeX listings

Typing inline code such as `E = m c^2 \neq \pi \neq \frac{3}{14}` is useful, as is showing use cases such as the following one.

```
Formatted \LaTeX\ code is great: $E = m c^2$ or $pi \neq \frac{3}{14}$.
```

Formatted LaTeX code is great: $E = mc^2$ or $pi \neq \frac{3}{14}$.

There's also a less invasive side-by-side mode. Nice! No ?

```
Formatted \LaTeX\ code is great:        \\
$E = m c^2$ or $\pi \neq \frac{3}{14}$.
```

Formatted LaTeX code is great:
$E = mc^2$ or $\pi \neq \frac{3}{14}$.

III. Highlighting, versioning and dating

1. tdocexa, tdocrem

1.7.0
2024-12-04

In the flow of the text, it is always useful to be able to provide examples and comments to supplement the main content.

Example III.1. *What to say[1]? I don't know, but it's nice. No ?*

Remark III.2. *What to say[2]? I don't know, but it's nice. No ?*

2. tdocnote, tdoctip...

Depending on the context of use, it is sometimes necessary to be able to highlight content by indicating its degree of importance.

Note.

What to say[a]? I don't know, but it's nice. No ?

[a] Let's not forget the footnotes...

Tip.

What to say? I don't know, but it's nice. No ?

Important.

What to say? I don't know, but it's nice. No ?

Caution.

What to say? I don't know, but it's nice. No ?

Warning.

What to say? I don't know, but it's nice. No ?

[1] Let's not forget the footnotes...
[2] Let's not forget the footnotes...

3. tdocbreak, tdocfix...

A new demonstration section...

Todo.

- A gallery would be welcome...

In a change log, it is important to visualise the types of changes clearly. This makes it easier for the user to read!

Break.

- Infos...

Fix.

- Infos...

New.

- Infos...

Problem.

- Infos...

Technical information.

- Infos...

Update.

- Infos...

Todo.

- Infos...

The "dark" theme

I. Liens

A very big link, but at least we can see it.

II. LaTeX listings

Typing inline code such as `E = m c^2 \neq \pi \neq \frac{3}{14}` is useful, as is showing use cases such as the following one.

```
Formatted \LaTeX\ code is great: $E = m c^2$ or $pi \neq \frac{3}{14}$.
```

Formatted LaTeX code is great: $E = mc^2$ or $pi \neq \frac{3}{14}$.

There's also a less invasive side-by-side mode. Nice! No ?

```
Formatted \LaTeX\ code is great:        \\
$E = m c^2$ or $\pi \neq \frac{3}{14}$.
```

Formatted LaTeX code is great:
$E = mc^2$ or $\pi \neq \frac{3}{14}$.

III. Highlighting, versioning and dating

1. tdocexa, tdocrem

1.7.0
2024-12-04

In the flow of the text, it is always useful to be able to provide examples and comments to supplement the main content.

Example III.1. *What to say[1]? I don't know, but it's nice. No ?*

Remark III.2. *What to say[2]? I don't know, but it's nice. No ?*

2. tdocnote, tdoctip...

Depending on the context of use, it is sometimes necessary to be able to highlight content by indicating its degree of importance.

Note.

What to say[a]? I don't know, but it's nice. No ?

[a] Let's not forget the footnotes...

Tip.

What to say? I don't know, but it's nice. No ?

Important.

What to say? I don't know, but it's nice. No ?

Caution.

What to say? I don't know, but it's nice. No ?

Warning.

What to say? I don't know, but it's nice. No ?

[1] Let's not forget the footnotes...
[2] Let's not forget the footnotes...

3. tdocbreak, tdocfix...

A new demonstration section...

Todo.

- A gallery would be welcome...

In a change log, it is important to visualise the types of changes clearly. This makes it easier for the user to read!

Break.

- Infos...

Fix.

- Infos...

New.

- Infos...

Problem.

- Infos...

Technical information.

- Infos...

Update.

- Infos...

Todo.

- Infos...

The "draft" theme

I. Liens

A very big link, but at least we can see it.

II. LaTeX listings

Typing inline code such as `E = m c^2 \neq \pi \neq \frac{3}{14}` is useful, as is showing use cases such as the following one.

```
Formatted \LaTeX\ code is great: $E = m c^2$ or $pi \neq \frac{3}{14}$.
```

Formatted LaTeX code is great: $E = mc^2$ or $pi \neq \frac{3}{14}$.

There's also a less invasive side-by-side mode. Nice! No ?

```
Formatted \LaTeX\ code is great:        \\
$E = m c^2$ or $\pi \neq \frac{3}{14}$.
```

Formatted LaTeX code is great:
$E = mc^2$ or $\pi \neq \frac{3}{14}$.

III. Highlighting, versioning and dating

1. tdocexa, tdocrem

1.7.0
2024-12-04

In the flow of the text, it is always useful to be able to provide examples and comments to supplement the main content.

Example III.1. *What to say[1]? I don't know, but it's nice. No ?*

Remark III.2. *What to say[2]? I don't know, but it's nice. No ?*

2. tdocnote, tdoctip...

Depending on the context of use, it is sometimes necessary to be able to highlight content by indicating its degree of importance.

Note III.3. *What to say[3]? I don't know, but it's nice. No ?*

Tip III.4. *What to say? I don't know, but it's nice. No ?*

Important III.5. *What to say? I don't know, but it's nice. No ?*

Caution III.6. *What to say? I don't know, but it's nice. No ?*

Warning III.7. *What to say? I don't know, but it's nice. No ?*

3. tdocbreak, tdocfix...

[Init] A new demonstration section...

Todo.

- A gallery would be welcome...

In a change log, it is important to visualise the types of changes clearly. This makes it easier for the user to read!

Break.

- Infos...

Fix.

- Infos...

New.

- Infos...

Problem.

- Infos...

Technical information.

- Infos...

Update.

- Infos...

Todo.

- Infos...

[1] Let's not forget the footnotes...
[2] Let's not forget the footnotes...
[3] Let's not forget the footnotes...