Le package `tutodoc` - Documentation de type tutoriel

Christophe BAL

26 sept. 2024 - Version 1.3.1

Résumé

Le package `tutodoc`[1] est utilisé par son auteur pour produire de façon sémantique des documentations de packages et de classes LaTeX dans un style de type tutoriel[2], et avec un rendu sobre pour une lecture sur écran.

Deux points importants à noter.

- Ce package impose un style de mise en forme. Dans un avenir plus ou moins proche, `tutodoc` sera sûrement éclaté en une classe et un package.
- Cette documentation est aussi disponible en anglais.

Abstract.

The `tutodoc` package[3] *is used by its author to semantically produce documentation of LaTeX packages and classes in a tutorial style*[4]*, 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.*

1. Le nom vient de « ***tuto·rial-type doc·umentation*** » qui se traduit en « *documentation de type tutoriel* ».
2. L'idée est de produire un fichier PDF efficace à parcourir pour des besoins ponctuels. C'est généralement ce que l'on attend d'une documentation liée au codage.
3. The name comes from « ***tuto·rial-type doc·umentation*** ».
4. 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.

Table des matières

I. Mises en forme générales imposées

1. Géométrie de la page

Le package `geometry` est chargé avec les réglages suivants.

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

2. Titre et table des matières

Les packages `titlesec` et `tocbasic` sont réglés comme suit.

```
\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. Liens dynamiques

Le package `hyperref` est importé en coulisse avec les réglages ci-dessous.

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

II. Choisir la langue au chargement du package

La présente documentation utilise le français via `\usepackage[lang = french]{tutodoc}` . Pour le moment, on a juste les deux choix suivants.

1. `english` est la valeur par défaut.
2. `french` est pour « *français* » en anglais.

Note.

Les noms des langues sont ceux proposés par le package `babel`.

III. Cela veut dire quoi en « *anglais* »

Penser aux non-anglophones est bien, même si ces derniers se font de plus en plus rares.

```
Cool et top signifient \tdocinEN*{cool} et \tdocinEN{top}.
```

Cool et top signifient « *cool* » et « *top* » en anglais.

La macro `\tdocinEN` et sa version étoilée s'appuient sur `\tdocquote` : par exemple, « *sémantique* » s'obtient via `\tdocquote{sémantique}` .

Note.

Le texte « en anglais » est traduit dans la langue indiquée lors de l'importation de `tutodoc`.

IV. Mettre en avant du contenu

Note.

Les environnements présentés dans cette section[a] *ajoutent un court titre indiquant le type d'informations fournies. Ce court texte sera toujours traduit dans la langue indiquée lors du chargement du package `tutodoc`.*

a. La mise en forme provient du package `keytheorems`.

1. Du contenu dans le flot de la lecture

Important.

Tous les environnements présentés dans cette section partagent le même compteur.

i. Des exemples

Des exemples numérotés, si besoin, s'indiquent via `\begin{tdocexa}` ... `\end{tdocexa}` qui propose un argument optionnel pour ajouter un mini-titre. Voici deux usages possibles.

```
\begin{tdocexa}
    Un exemple...
\end{tdocexa}

\begin{tdocexa}[Mini titre]
    Utile ?
\end{tdocexa}
```

Exemple 1. *Un exemple...*

Exemple 2 (Mini titre)**.** *Utile ?*

Important.

La numérotation des exemples est remise à zéro dès qu'une section de niveau au moins égale à une `\subsubsection` *est ouverte.*

Astuce.

Il peut parfois être utile de revenir à la ligne dès le début du contenu. Le code suivant montre comment faire (ce tour de passe-passe reste valable pour l'environnement `tdocrem` *présenté juste après). Noter au passage que la numérotation suit celle de l'exemple précédent comme souhaité.*

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

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

Exemple 3.

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

ii. Des remarques

Tout se passe via `\begin{tdocrem}` ... `\end{tdocrem}` avec un fonctionnement identique à l'environnement `tdocexa` comme le montre l'exemple suivant.

```
\begin{tdocrem}
    Juste une remarque...
\end{tdocrem}

\begin{tdocrem}[Mini titre]
    Utile ?
\end{tdocrem}
```

Remarque 4. *Juste une remarque...*

Remarque 5 (Mini titre)**.** *Utile ?*

2. Du contenu tape-à-l'oeil

Note.

Les icônes sont obtenues via le package `fontawesome5`, et la gestion de l'espacement avec le texte est faite par la macro `\tdocicon`.

i. Une astuce

L'environnement `tdoctip` sert à donner des astuces. Voici comment l'employer.

```
\begin{tdoctip}
    Une astuce.
\end{tdoctip}

\begin{tdoctip}[Mini titre]
    Utile ?
\end{tdoctip}
```

Astuce.

Une astuce.

Astuce (Mini titre).

Utile ?

Note.

Les couleurs sont fournies par les macros développables `\tdocbackcolor` et `\tdocdarkcolor` qui admettent les codes suivants et attendent une couleur au format `xcolor` comme argument.

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

ii. Note informative

L'environnement `tdocnote` sert à mettre en avant des informations utiles. Voici comment l'utiliser.

```
\begin{tdocnote}
    Un truc utile à vous dire...
\end{tdocnote}

\begin{tdocnote}[Mini titre]
    Utile ?
\end{tdocnote}
```

Note.

Un truc utile à vous dire...

Note (Mini titre).

Utile ?

iii. Un truc important

L'environnement `tdocimp` permet d'indiquer quelque chose d'important mais sans danger.

```
\begin{tdocimp}
    Un truc important sans danger.
\end{tdocimp}

\begin{tdocimp}[Mini titre]
    Utile ?
\end{tdocimp}
```

Important.

Un truc important sans danger.

Important (Mini titre).

Utile ?

iv. Avertir d'un point très délicat

L'environnement `tdoccaut` sert à indiquer un point délicat à l'utilisateur. Voici comment l'employer.

```
\begin{tdoccaut}
    Prudence, prudence...
\end{tdoccaut}

\begin{tdoccaut}[Mini titre]
    Utile ?
\end{tdoccaut}
```

Mise en garde.

Prudence, prudence...

Mise en garde (Mini titre).

Utile ?

v. Avertir d'un danger

L'environnement `tdocwarn` sert à avertir l'utilisateur d'un piège à éviter. Voici comment l'employer.

```
\begin{tdocwarn}
    Evitez les dangers...
\end{tdocwarn}

\begin{tdocwarn}[Mini titre]
    Utile ?
\end{tdocwarn}
```

Avertissement.

Evitez les dangers...

Avertissement (Mini titre).

Utile ?

V. Indiquer des packages, des classes, des macros ou des environnements

Voici ce qu'il est possible de taper de façon sémantique.

```
\tdoccls{maclasse} sert à...

\tdocpack{monpackage} est pour...

\tdocmacro{unemacro} permet de...

\tdocenv{env} produit...

Juste \tdocenv*{env}...

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

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

`maclasse` sert à...
`monpackage` est pour...
`\unemacro` permet de...
`\begin{env}...\end{env}` produit...
Juste `env`...
`\begin{env}[opt1]<opt2>...\end{env}`
Enfin `env`...

Remarque 1. *Contrairement à `\tdocinlatex`, les macros `\tdocenv` et `\tdocenv*` ne colorent pas le texte produit. De plus, `\tdocenv{monenv}` produit `\begin{monenv}...\end{monenv}` avec des espaces afin d'autoriser des retours à la ligne si besoin.*

Avertissement.

L'argument optionnel de la macro `\tdocenv` est copié-collé[a] *lors du rendu. Ceci peut donc parfois nécessiter d'utiliser des accolades protectrices comme dans l'exemple ci-dessus.*

a. Se souvenir que tout est possible ou presque dorénavant.

VI. Origine d'un préfixe ou d'un suffixe

Pour expliquer les noms retenus, rien de tel que d'indiquer et expliciter les courts préfixes et suffixes employés. Ceci se fait facilement comme suit.

```
\tdocpre{sup} est relatif à...

\tdocprewhy{sup.erbe} signifie...

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

sup est relatif à...
sup·**erbe** signifie...
*sup·**er** pour...*

Remarque 1. *Le choix du point pour scinder un mot permet d'utiliser des mots avec un tiret comme dans* `\tdocprewhy{ca.sse-brique}` *qui donne* ***ca·sse-brique****.*

VII. Un rendu en situation réelle

Il est parfois utile d'obtenir directement le rendu d'un code dans la documentation. Ceci nécessite que ce type de rendu soit dissociable du texte donnant des explications.

1. Avec une bande colorée

Exemple 1 (Avec les textes par défaut)**.** *Il peut être utile de montrer un rendu réel directement dans un document.*[5] *Ceci se tape via* `\begin{tdocshowcase} ... \end{tdocshowcase}` *comme suit.*

```
\begin{tdocshowcase}
    \bfseries Un peu de code \LaTeX.

    \bigskip

    \emph{\large Fin de l'affreuse démo.}
\end{tdocshowcase}
```

On obtient alors le rendu suivant.[6]

Début du rendu réel

Un peu de code LaTeX.

Fin de l'affreuse démo.

Fin du rendu réel

Remarque 2. *Voir la section 4. page 12 pour obtenir facilement un code suivi de son rendu réel comme dans l'exemple précédent.*

Note.

Les textes explicatifs s'adaptent à la langue choisie lors du chargement de `tutodoc`*.*

Exemple 3 (Changer la couleur et/ou les textes par défaut)**.**

```
\begin{tdocshowcase}[before = Mon début,
                     after  = Ma fin à moi,
                     color  = red]
    Bla, bla, bla, bla, bla, bla, bla, bla, bla, bla, bla, bla, bla...
\end{tdocshowcase}
```

Ceci produira ce qui suit.

5. Typiquement lorsque l'on fait une démo.
6. En coulisse, la bande est créée sans effort grâce au package `clrstrip`.

Mon début

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

Ma fin à moi

Note.

Vous avez sûrement remarqué que le rouge sert de base pour obtenir les couleurs utilisées.

- *La couleur de fond est fournie par* `\tdocbackcolor`.
- *La couleur des titres et des lignes est fournie par* `\tdocdarkcolor`.

Ces macros développables et à un seul argument, la couleur choisie, admettent les codes suivants.

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

Il faut également savoir qu'en coulisse, la macro `\tdocruler` *est utilisée.*

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

Un pseudo-titre décoré

Avertissement.

Avec les réglages par défaut, si le code LaTeX à mettre en forme commence par un crochet ouvrant, il faudra user de `\string` *comme dans l'exemple suivant.*

```
\begin{tdocshowcase}
    \string[Cela fonctionne...]
\end{tdocshowcase}
```

Ceci produira ce qui suit.

Début du rendu réel

[Cela fonctionne...]

Fin du rendu réel

2. Sans bande colorée

Le rendu de `\begin{tdocshowcase}` ...`\end{tdocshowcase}` avec une bande colorée peut ne pas convenir, ou parfois ne pas être acceptable malgré le travail fait par `clrstrip`. Il est possible de ne pas utiliser une bande colorée comme nous allons le voir tout de suite.

Exemple 1. *L'emploi de* `\begin{tdocshowcase}[nostripe]` ...`\end{tdocshowcase}` *demande de ne pas faire appel à* `clrstrip`. *Voici un exemple d'utilisation.*

```
\begin{tdocshowcase}[nostripe]
    Bla, bla, bla, bla, bla, bla, bla, bla, bla, bla, bla, bla, bla...
\end{tdocshowcase}
```

Ceci produira ce qui suit.

Début du rendu réel

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

Fin du rendu réel

Exemple 2 (Changer la couleur et/ou les textes par défaut)**.**

```
\begin{tdocshowcase}[nostripe,
                     before = Mon début,
                     after  = Ma fin à moi,
                     color  = green]
    Bla, bla, bla, bla, bla, bla, bla, bla, bla, bla, bla, bla, bla...
\end{tdocshowcase}
```

Ceci produira ce qui suit.

Mon début

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

Ma fin à moi

3. En important le code LaTeX

Pour obtenir des rendus en important le code depuis un fichier externe, au lieu de le taper, il suffit d'employer la macro `\tdocshowcaseinput` dont l'option reprend la syntaxe de celle de `\begin{tdocshowcase} ...\end{tdocshowcase}` et l'argument obligatoire correspond au chemin du fichier.

Exemple 1. *Ce qui suit a été obtenu via* `\tdocshowcaseinput{external.tex}` .

Début du rendu réel

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

Fin du rendu réel

Quant à `\tdocshowcaseinput[color = orange]{external.tex}` *, ceci produira le changement de couleur observable ci-après.*

Début du rendu réel

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

Fin du rendu réel

VIII. Cas d'utilisation en LaTeX

Documenter un package ou une classe se fait efficacement via des cas d'utilisation montrant à la fois du code et le résultat correspondant.[7]

Mise en garde.

La version 3 de `minted` *ne peut pas être prise en compte pour le moment car elle comporte des bugs : voir* `https://github.com/gpoore/minted/issues/401` *. On force donc l'usage de la version 2 de* `minted`*.*

1. Codes « *en ligne* »

La macro `\tdocinlatex`[8] permet de taper du code en ligne via un usage similaire à `\verb`. Voici des exemples d'utilisation.

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

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

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

7. La mise en forme des codes se fait via le package `minted`.
8. Le nom de la macro `\tdocinlatex` vient de « ***in·line*** *LaTeX* » soit « *LaTeX en ligne* » en anglais.

Note.

La macro `\tdocinlatex` *est utilisable dans une note de pied de page : voir ci-dessous.*[a] *De plus, une couleur de fond est volontairement utilisée pour subtilement faire ressortir les codes* `\LaTeX` .

a. `$minted = TOP$` a été tapé `\tdocinlatex+$minted = TOP$+` dans cette note de bas de page..

2. Codes tapés directement

Exemple 1 (Face à face). *Via* `\begin{tdoclatex}[sbs] ... \end{tdoclatex}`*, on affichera un code et son rendu côte à côte. Indiquons que* `sbs` *est pour « s·ide b·y s·ide » soit « côte à côte » en anglais. Considérons le code suivant.*

```
\begin{tdoclatex}[sbs]
    $A = B + C$
\end{tdoclatex}
```

Ceci produira ce qui suit.

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

$A = B + C$

Exemple 2 (À la suite). `\begin{tdoclatex} ... \end{tdoclatex}` *produit le résultat suivant qui correspond à l'option par défaut* `std` .[9]

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

$A = B + C$

Exemple 3 (Juste le code). *Via* `\begin{tdoclatex}[code] ... \end{tdoclatex}`*, on aura juste le code comme ci-après.*

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

Avertissement.

Avec la mise en forme par défaut, si le code commence par un crochet ouvrant, il faudra indiquer explicitement l'option par défaut. Considérons le code suivant.

```
\begin{tdoclatex}[std]
    [Étrange... Ou pas !]
\end{tdoclatex}
```

Ceci produira ce qui suit.

```
[Étrange... Ou pas !]
```

[Étrange... Ou pas !]

Une autre méthode consiste à utiliser la primitive `\string`*. Considérons le code suivant.*

```
\begin{tdoclatex}
    \string[Étrange... Ou pas !]
\end{tdoclatex}
```

Ceci produira ce qui suit.

9. `std` fait référence au comportement « *standard* » de `tcolorbox` vis à vis de la librairie `minted`.

```
[Étrange... Ou pas !]
```

[Étrange... Ou pas !]

3. Codes importés

Pour les codes suivants, on considère un fichier de chemin relatif `examples-listing-xyz.tex`, et ayant le contenu suivant.

```
% Juste une démo.
$x y z = 1$
```

La macro `\tdoclatexinput`, présentée ci-dessous, attend le chemin d'un fichier et propose les mêmes options que l'environnement `tdoclatex`.

Exemple 1 (Face à face)**.**

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

Ceci produit la mise en forme suivante.

```
% Juste une démo.
$x y z = 1$
```

$xyz = 1$

Exemple 2 (À la suite)**.**

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

Ceci produit la mise en forme suivante où l'option employée par défaut est `std`*.*

```
% Juste une démo.
$x y z = 1$
```

$xyz = 1$

Exemple 3 (Juste le code)**.**

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

Ceci produit la mise en forme suivante.

```
% Juste une démo.
$x y z = 1$
```

4. Codes importés et mis en situation

Exemple 1 (Mise en situation)**.** *Ce qui suit s'obtient via* `\tdoclatexshow{examples-listing-xyz.tex}`*.*

Début du rendu dans cette doc.

```
% Juste une démo.
$x y z = 1$
```

Ceci donne :

Début du rendu réel

$xyz = 1$

Fin du rendu réel

Fin du rendu dans cette doc.

Note.

Les textes par défaut tiennent compte de la langue choisie lors du chargement du package `tutodoc`*.*

Exemple 2 (Changer le texte explicatif)**.** *Via la clé* `explain` *, on peut utiliser un texte personnalisé. Ainsi,* `\tdoclatexshow[explain = Voici le rendu réel.]{examples-listing-xyz.tex}` *produira ce qui suit.*

Début du rendu dans cette doc.

```
% Juste une démo.
$x y z = 1$
```

Voici le rendu réel.

Début du rendu réel

$xyz = 1$

Fin du rendu réel

Fin du rendu dans cette doc.

Exemple 3 (Les options disponibles)**.** *En plus du texte explicatif, il est aussi possible d'utiliser toutes les options de l'environnement* `tdocshowcase`*, voir* *VII. page 8.* *Voici un exemple illustrant ceci.*

```
\tdoclatexshow[explain = Ce qui vient est coloré...,
               before  = Rendu ci-après.,
               after   = Rendu fini.,
               color   = orange]
               {examples-listing-xyz.tex}
```

Ceci va produire ce qui suit.

Début du rendu dans cette doc.

```
% Juste une démo.
$x y z = 1$
```

Ce qui vient est coloré...

Rendu ci-après.

$xyz = 1$

Rendu fini.

Fin du rendu dans cette doc.

IX. Indiquer les changements

Afin de faciliter le suivi d'un package, il est indispensable de fournir un historique indiquant les changements effectués lors de la publication d'une nouvelle version.

1. À quel moment ?

On peut au choix dater quelque chose, ou bien le versionner, dans ce second cas le numéro de version pourra éventuellement être daté.

Exemple 1 (Dater des nouveautés)**.** *La macro* `\tdocdate` *permet d'indiquer une date dans la marge comme dans l'exemple suivant.*

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

\medskip % ATTENTION ! Ceci évite le chevauchement.

\tdocdate{2023-09-24}

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

\medskip % ATTENTION ! Ceci évite le chevauchement.

\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...
```

Ceci donne :

Début du rendu réel

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...

Fin du rendu réel

Exemple 2 (Versionner des nouveautés en les datant éventuellement)**.** *Associer un numéro de version à une nouveauté se fait via la macro* `\tdocversion`*, la couleur et la date étant des arguments optionnels.*

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

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

\bigskip % ATTENTION ! Ceci évite le chevauchement.

\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...
```

Ceci donne :

Début du rendu réel

10.2.0-beta
01/12/2023
10.2.0-alpha

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

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...

Fin du rendu réel

Important.

1. *Les macros `\tdocdate` et `\tdocversion` nécessitent deux compilations.*
2. *Comme la langue indiquée pour cette documentation est le français, la date dans le rendu final est au format `JJ/MM/AAAA` alors que dans le code celle-ci devra toujours être saisie au format anglais `AAAA-MM-JJ`.*

Avertissement.

Seul l'emploi du format numérique `YYYY-MM-DD` est vérifié, [a] *et ceci est un choix ! Pourquoi cela ? Tout simplement car dater et versionner des explications devrait se faire de façon semi-automatisée afin d'éviter tout bug humain.*

a. Techniquement, vérifier la validité d'une date, via LaTeX3, ne présente pas de difficulté.

2. Quoi de neuf ?

`tutodoc` propose la macro `\tdocstartproj` et différents environnements pour indiquer rapidement et clairement ce qui a été fait lors des derniers changements. [10]

Note.

Les icônes sont obtenues via le package `fontawesome5`, et la gestion de l'espacement avec le texte est faite par la macro `\tdocicon`.

Exemple 1 (Juste pour la toute première version).

```
\tdocstartproj{Première version du projet.}
```

Première version du projet.

Exemple 2 (Pour les nouveautés).

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

Nouveau.

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

Exemple 3 (Pour les mises à jour).

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

Mise à jour.

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

Exemple 4 (Pour les bifurcations).

10. L'utilisateur n'a pas besoin de tous les détails techniques.

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

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

Exemple 5 (Pour les problèmes).

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

Problème.
- *Info 1...*
- *Info 2...*

Exemple 6 (Pour les réparations).

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

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

Exemple 7 (Thématiques aux choix avec une icône).

```
\begin{tdoctopic}{Ne pas
↪  regarder}[\faEyeSlash]
% Une icône venant de fontawesome5.
    \item Info 1...
    \item Info 2...
\end{tdoctopic}
```

Ne pas regarder.
- *Info 1...*
- *Info 2...*

Exemple 8 (Thématiques aux choix sans icône).

```
\begin{tdoctopic}{La fin des icônes}
% Ici le point s'impose.
    \item Info 1...
    \item Info 2...
\end{tdoctopic}
```

La fin des icônes.
- *Info 1...*
- *Info 2...*

Note.

L'utilisation d'une icône suivie du bon espacement juste avant le texte se fait via la macro `\tdocicon`. *Par exemple,* `\tdocicon{\faBed}{Fatigué}` *produit* Fatigué.

X. Décorations

Finissons cette documentation avec de petits outils de mise en forme pouvant rendre de grands services.

```
Bla, bla, bla...

\tdocsep % Pratique pour délimiter.

Ceci fonctionne avec des énumérations.

\begin{itemize}
    \item Point souligné.

    \item Autre chose utile.
\end{itemize}

\tdocsep % Un comportement uniforme.

Ble, ble, ble...

Bli, bli, bli...

\tdocxspace % Espace subtile
            % mais utile.

Blo, blo, blo...

Blu, blu, blu...
```

Bla, bla, bla...

Ceci fonctionne avec des énumérations.

- Point souligné.
- Autre chose utile.

Ble, ble, ble...
Bli, bli, bli...
Blo, blo, blo...
Blu, blu, blu...

XI. Historique

1.3.1
26/09/2024

Nouveau.

- Version étoilée de `\tdocenv` pour n'avoir que le nom de l'environnement.

1.3.0
25/09/2024

Problème.

- La version 3 de `minted` ne peut pas être prise en compte pour le moment car elle comporte des bugs : voir https://github.com/gpoore/minted/issues/401. On force donc l'usage de la version 2 de `minted`.

Bifurcation.

- L'environnement `tdocimportant` a été renommé `tdocimp` pour une saisie simplifiée.

Nouveau.

- Journal des changements : les environnements proposés utilisent des icônes.
- Mise en avant de contenus : des cadres colorés avec des icônes sont proposés pour les environnements suivants.
  1. `tdoccaution`
  2. `tdocimp`
  3. `tdocnote`
  4. `tdoctip`
  5. `tdocwarn`

1.2.0-a
23/08/2024

Mise à jour.

- `\tdocversion`
  1. Le numéro de version est au-dessus de la date.
  2. L'espacement est mieux géré lorsque la date est absente.

Réparation.

- Mise en avant de contenus : les traductions françaises de « *caution* » et « *danger* » étaient erronées.

1.1.0
06/01/2024

Nouveau.

- Journal des changements : deux nouveaux environnements.
  1. `\begin{tdocbreak}...\end{tdocbreak}` pour les « *bifurcations* », soit les modifications non rétro-compatibles.

2. `\begin{tdocprob}...\end{tdocprob}` pour les problèmes repérés.

- `\tdocinlatex` : un jaune léger est utilisé comme couleur de fond.

1.0.1
08/12/2023

Réparation.

- `\tdocenv` : l'espacement est maintenant correct, même si le paquet `babel` n'est pas chargé avec la langue française.
- `\begin{tdocshowcase}[nostripe]...\end{tdocshowcase}` : les sauts de page autour des lignes « *cadrantes* » devraient être rares dorénavant.

1.0.0
29/11/2023

Première version publique du projet.