

La classe tutodoc

Documentation de type tutoriel

Christophe, BAL

18 déc. 2024 - Version 1.7.1

La classe `tutodoc`¹ est utilisée par son auteur pour produire de façon sémantique des documentations de packages et de classes L^AT_EX dans un style de type tutoriel² avec un rendu sobre pour une lecture sur écran.

Remarque : cette documentation est aussi disponible en anglais.

Abstract.

The `tutodoc` class³ is used by its author to semantically produce documentation of L^AT_EX packages and classes in a tutorial style⁴ with a sober rendering for reading on screen.

Remark : this documentation is also available in French.

Derniers changements

Réparation.

- Documentation : les références aux outils pour indiquer des changements ont été incorrectement écrites comme caractéristiques des contenus colorés mis en avant.

Bifurcation.

- La macro `\tdocenv` et sa version étoilée ne proposent plus d'option.
- Cas d'utilisation L^AT_EX : la présentation par défaut est plus sobre, et des options permettent d'ajouter les lignes cadrantes, ou la bande colorée. Voir juste après.

Nouveau.

- Mise en forme de codes informatiques en plus de ceux spécifiquement en L^AT_EX.
 1. Création de `\begin{tdoccode} ... \end{tdoccode}` et de `\tdoccodein`.
 2. Pour les macros pour du code en ligne, et les environnements pour des blocs de code, des options de type `minted` s'indiquent à l'intérieur de crochet de manière traditionnelle : `[options minted]`.
 3. Pour les environnements pour des blocs de code, des options de type `tcolorbox` s'indiquent à l'intérieur de chevrons : `<options tcolorbox>`.
 4. La nouvelle macro `\tdoctcb` permet d'utiliser des raccourcis pour les styles `tcolorbox` régulièrement utilisés.
- Documentation : une nouvelle section présente les outils de mise en forme de codes informatiques autres que les cas d'utilisation de L^AT_EX.

Mise à jour.

- Les sous-sous-sections sont numérotées en lettres minuscules.
- Thèmes.
 1. Moins d'espace consommé.
 2. Les ombres ont une meilleure coloration.
 3. Pour tous les thèmes sauf le `draft`, le rayon des arcs des coins des cadres passe de `.75mm` à `.2pt`.
 4. Cas d'utilisation de L^AT_EX : avec le thème `color`, la couleur du fond passe de `yellow!4` à `gray!5`.
 5. Dernières modifications : avec le thème `dark`, le texte `[Init]` produit par la macro `\tdocstartproj` utilise la même police que celle des titres des environnements pour indiquer des changements.

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 for a coding documentation.

Table des matières

I. Dépendances	4
II. Réglages généraux	4
1. Taille de la police et géométrie de la page	4
2. Titres et table des matières	4
3. Liens dynamiques	4
III. Quelle langue est utilisée par la classe <code>tutodoc</code> ?	4
IV. Cela veut dire quoi en « <i>anglais</i> »	5
V. Choisir son thème	5
VI. Mettre en avant du contenu	5
1. Du contenu dans le flot de la lecture	5
a. Des exemples	5
b. Des remarques	6
2. Du contenu tape-à-l'oeil	6
a. Une astuce	6
b. Note informative	7
c. Un truc important	7
d. Avertir d'un point très délicat	7
e. Avertir d'un danger	8
VII. Indiquer des packages, des classes, des macros ou des environnements	8
VIII. Origine d'un préfixe ou d'un suffixe	8
IX. Un rendu en situation réelle	8
1. Un rendu minimaliste par défaut	8
2. Avec des lignes cadrantes	9
3. Avec une bande colorée	10
4. En important le code <code>L^AT_EX</code>	10
X. Cas d'utilisation en <code>L^AT_EX</code>	11
1. Codes « <i>en ligne</i> »	11
2. Codes tapés directement	11
3. Codes importés	13
4. Codes importés et mis en situation	14
XI. Présenter du code informatique	14
1. Codes « <i>en ligne</i> »	15
2. Codes tapés directement	15
3. Codes importés	16
XII. Indiquer les changements	17
1. À quel moment ?	17
2. Quoi de neuf ?	18
a. La sobriété avant tout	19
b. De la couleur si besoin	20
3. Le quoi et le quand	20
XIII. Décoration	20
XIV. Contribuer	21
1. Compléter les traductions	21
a. Les dossiers <code>fr</code> et <code>en</code>	22
b. Le dossier <code>changes</code>	22
c. Le dossier <code>status</code>	22
d. Les fichiers <code>README.md</code> et <code>LICENSE.txt</code>	22
e. De nouvelles traductions	22
2. Améliorer le code source	23

XV. Historique

23

Annexe – Galerie des thèmes

27

I. Dépendances

tutodoc admet les dépendances suivantes (les dates entre parenthèses sont celles des versions utilisées lors des derniers tests).

- `scartcl.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)
- `clstrip.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. Réglages généraux

1. Taille de la police et géométrie de la page

La classe `scartcl` est chargée via l'option `fontsize = 10pt`, et le package `geometry` gère les dimensions de la page.

Avertissement.

Les macros pour dater et versionner présentées dans la section XII page 17 ont besoin de réglages fixes pour la géométrie des pages et la taille de la police.

2. Titres et table des matières

Les réglages choisis sont directement visibles dans cette documentation.

3. Liens dynamiques

Le package `hyperref` est importé, si ce n'est pas fait, et les réglages choisis portent juste sur les couleurs des liens relatifs aux citations, aux fichiers, aux liens internes, et enfin aux `url` (ces couleurs dépendent du thème choisi).

III. Quelle langue est utilisée par la classe tutodoc ?

Cette documentation charge le package `babel` via `\usepackage[french]{babel}` un package que `tutodoc` ne charge pas. En revanche, la classe `tutodoc` identifie `fr` comme la langue principale utilisée par `babel`.⁵ Comme cette langue fait partie de la liste des langues prises en compte, voir ci-dessous, la classe `tutodoc` produira les effets attendus.

- `en` : anglais.
- `es` : espagnol.
- `fr` : français.

Note.

Les packages `babel` et `polyglossia` sont pris en compte.

Mise en garde.

Si le choix de la langue principale n'est pas faite dans le préambule, le mécanisme employé échouera avec des effets de bord non voulus (voir l'avertissement qui suit).

Avertissement.

Lorsqu'une langue n'est pas prise en compte par `tutodoc`, un message d'avertissement est émis, et l'anglais est alors choisi comme langue vis-à-vis de `tutodoc`.

⁵. Techniquement, on utilise `\BCPdata{language}` qui renvoie une langue au format court.

IV. 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 détectée par `tutodoc`.

Dans le contenu entre guillemets, les règles linguistiques anglaises sont respectées comme le montre l'exemple suivant.

Test: <code>\tdocquote{OK?}</code> \\	Test : « OK ? »
<code>\tdocinEN{Test: \tdocquote{OK?}}</code>	« Test: "OK?" » en anglais

V. Choisir son thème

Pour modifier la mise en forme générale, la classe `tutodoc` propose l'option `theme = <choix>` où `<choix>` peut prendre les valeurs suivantes.

- `bw` : un thème de type noir-et-blanc avec certaines nuances de gris.
- `color` : un thème coloré : c'est la valeur par défaut.
- `dark` : un thème sombre idéal pour se reposer les yeux.
- `draft` : un thème pour une impression papier à la recherche d'erreurs de contenu pas forcément simples à débusquer devant un écran.

Note.

À la fin de ce document, après l'historique des changements, vous trouverez une galerie de cas d'utilisation de ces différents thèmes : se rendre à l'annexe page 27.

VI. 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 repérée par la classe `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 qui sera remis à zéro dès que `\section` est employée.

a. Des exemples

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

<pre>\begin{tdocexa} Un exemple... \end{tdocexa} \begin{tdocexa}[Mini titre] Utile ? \end{tdocexa}</pre>	<p>Exemple VI.1. <i>Un exemple...</i></p> <p>Exemple VI.2 (Mini titre). <i>Utile ?</i></p>
---	--

 **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.

<pre>\begin{tdocexa} \leavevmode \begin{enumerate} \item Point 1. \item Point 2. \end{enumerate} \end{tdocexa}</pre>	<p>Exemple VI.3.</p> <ol style="list-style-type: none"> 1. <i>Point 1.</i> 2. <i>Point 2.</i>
---	--

b. Des remarques

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

<pre>\begin{tdocrem} Juste une remarque... \end{tdocrem} \begin{tdocrem}[Mini titre] Utile ? \end{tdocrem}</pre>	<p>Remarque VI.4. <i>Juste une remarque...</i></p> <p>Remarque VI.5 (Mini titre). <i>Utile ?</i></p>
---	--

2. Du contenu tape-à-l'oeil

 **Note.**

La mise en forme proposée ici est celle par défaut, mais d'autres sont possible en changeant de thème : voir la galerie de cas d'utilisation dans l'annexe page 27. Quant aux icônes, elles sont obtenues via le package `fontawesome5` et la macro `\tdocicon` qui gère l'espacement vis-à-vis du texte.^a

a. Par exemple, `\fbox{\tdocicon{\faBed}{Fatigué}}` produit  Fatigué .

a. Une astuce

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

<pre>\begin{tdoctic} Une astuce. \end{tdoctic} \begin{tdoctic}[Mini titre] Utile ? \end{tdoctic}</pre>	<div style="background-color: #c8e6c9; padding: 5px; border: 1px solid #ccc; margin-bottom: 5px;">  Astuce. <i>Une astuce.</i> </div> <div style="background-color: #c8e6c9; padding: 5px; border: 1px solid #ccc;">  Astuce (Mini titre). <i>Utile ?</i> </div>
---	--

💡 Astuce.

Quelque fois, un contenu mis en avant peut se réduire à une liste. Dans ce cas, la mise en forme peut être améliorée comme suit où nous utilisons l'option `wide` du package `enumitem` qui est importé par cette documentation.

```
\begin{tdocticip}[Peu élégant]
  \begin{enumerate}
    \item Point 1.
    \item Point 2.
  \end{enumerate}
\end{tdocticip}
```

VERSUS.

```
\begin{tdocticip}[Plus élégant]
  \begin{enumerate}[wide]
    \item Point 1.
    \item Point 2.
  \end{enumerate}
\end{tdocticip}
```

💡 Astuce (Peu élégant).

1. Point 1.
2. Point 2.

VERSUS.

💡 Astuce (Plus élégant).

1. Point 1.
2. Point 2.

b. 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 ?

c. 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 ?

d. Avertir d'un point très délicat

L'environnement `tdoccaut` sert à pointer quelque chose de délicat. 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 ?

e. Avertir d'un danger

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

<pre>\begin{tdocwarn} Evitez les dangers... \end{tdocwarn} \begin{tdocwarn}[Mini titre] Utile ? \end{tdocwarn}</pre>	<div style="background-color: #f08080; padding: 5px; border: 1px solid #ccc;"> Avertissement.</div> <div style="background-color: #ffe6e6; padding: 5px; border: 1px solid #ccc;"><i>Evitez les dangers...</i></div>
	<div style="background-color: #f08080; padding: 5px; border: 1px solid #ccc;"> Avertissement (Mini titre).</div> <div style="background-color: #ffe6e6; padding: 5px; border: 1px solid #ccc;"><i>Utile ?</i></div>

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

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

<pre>\tdoccls{maclasse} sert à... \\ \tdocpack{monpackage} est pour... \\ \tdocmacro{unemacro} permet de... \\ \tdocenv{env} produit... \\ Juste \tdocenv*{env}...</pre>	<pre>maclasse sert à... monpackage est pour... \unemacro permet de... \begin{env}... \end{env} produit... Juste env...</pre>
--	--

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

VIII. 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.

<pre>\tdocpre{sup} est relatif à... \\ \tdocprewhy{sup.erbe} signifie... \\ \emph{\tdocprewhy{sup.er} pour...}</pre>	<pre>sup est relatif à... sup.erbe signifie... sup·er pour...</pre>
--	---

Remarque VIII.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*.

IX. 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. Un rendu minimaliste par défaut

Exemple IX.1. Il peut être utile de montrer un rendu réel directement dans un document.⁶ Ceci se tape via l'environnement `tdocshowcase` comme suit.

<pre>\begin{tdocshowcase} \bfseries Un peu de code <i>LaTeX</i>. \bigskip \emph{\large Fin de l'affreuse démo.} \end{tdocshowcase}</pre>
--

On obtient alors le rendu suivant qui est la combinaison d'un faible espacement vertical et d'une simple importation.

6. Typiquement lorsque l'on fait une démo.

Un peu de code \LaTeX .

Fin de l'affreuse démo.

Remarque IX.2. La section 4 page 14 explique comment obtenir, via la macro `\tdoclatexshow`, un code suivi de son rendu réel comme dans l'exemple précédent.

⚠ Avertissement.

Avec les paramètres par défaut, si le code à formater commence par un crochet ouvrant, il faut utiliser l'une des astuces suivantes.

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

Ceci produira ce qui suit.

[Cela fonctionne...]
OU.
[Cela fonctionne aussi...]

2. Avec des lignes cadrantes

Pour rendre plus visible le code \LaTeX mis en forme, on peut faire appel au style `rule` comme dans les exemple suivants.

Exemple IX.3. L'option `style = rule` permet d'obtenir ce qui suit où les textes ajoutés automatiquement s'adapteront à la langue repéré par `tutodoc`.

————— Début du rendu réel —————

Bla, bla...

————— Fin du rendu réel —————

Exemple IX.4 (Du texte et des couleurs modifiables). On peut obtenir facilement l'horreur suivante.

————— Mon début —————

Bla, bla...

————— Ma fin à moi —————

Voici le code qui a été employé.⁷

```
\begin{tdocshowcase}[style      = rule,  
                      col-stripe = red,  
                      col-text  = orange!75!black,  
                      before     = Mon début,  
                      after      = Ma fin à moi]  
  Bla, bla...  
\end{tdocshowcase}
```

i Note.

Dans l'exemple précédent, le texte utilise bien l'orange assombri proposé. Par contre, le rouge sert de base pour obtenir les couleurs utilisées pour les lignes cadrantes : les transformations utilisées dépendent du thème choisi.^a Il faut également savoir qu'en coulisse, la macro `\tdocruler` est employée, elle fonctionne comme suit.

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

————— Un pseudo-titre décoré —————

7. La section suivante va justifier le choix, a priori étrange, de `col-stripe` au lieu de `col-rule`.

a. Par exemple, les thèmes `bw` et `draft` ne tiennent pas compte de la clé `col-stripe`!

3. Avec une bande colorée

Il est des situations où l'on doit pouvoir clairement identifier un exemple de code `LATEX` mis en forme. Ceci est faisable comme le montrent les exemples suivants.⁸

Exemple IX.5. L'option `style = stripe` fournit ce qui suit.

```
————— Début du rendu réel —————  
Bla, bla...  
————— Fin du rendu réel —————
```

Exemple IX.6 (Du texte et des couleurs modifiables). On peut produire facilement une belle horreur comme celle qui arrive.

```
————— Mon début —————  
Bla, bla...  
————— Ma fin à moi —————
```

Voici le code qui a été employé.⁹

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

4. 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 l'environnement `tdocshowcase`, et l'argument obligatoire correspond au chemin du fichier. Voici des exemples d'utilisation.

Exemple IX.7 (Usage standard).

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

Ceci donne :
Blablobli, blablobli, blablobli, blablobli, blablobli, blablobli...

Exemple IX.8 (Avec des lignes cadrantes).

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

Ceci donne :
————— Début du rendu réel —————
Blablobli, blablobli, blablobli, blablobli, blablobli, blablobli...
————— Fin du rendu réel —————

Exemple IX.9 (Une bande colorée).

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

Ceci donne :

8. En coulisse, les bandes sont créées sans effort grâce au package `clrstrip`.
9. On comprend maintenant le choix de `col-stripe` au lieu de `col-rule`.

X. 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.¹⁰

1. Codes « en ligne »

Exemple X.1 (Usage standard). La macro `\tdoclatexin`¹¹ permet de taper du code en ligne via un usage similaire à `\verb`, ou bien comme une macro standard (voir la gestion des accolades dans le dernier cas ci-dessous). Voici des exemples d'utilisation.¹²

1: <code>\tdoclatexin/\$a^b = c\$ </code>		1: $a^b = c$
2: <code>\tdoclatexin+\tdoclatexin/\$a^b = c\$/+</code>		2: <code>\tdoclatexin/\$a^b = c\$ </code>
3: <code>\tdoclatexin{\tdoclatexin{\$a^b = c\$}}</code>		3: <code>\tdoclatexin{\$a^b = c\$}</code>

Exemple X.2 (Des options possibles). Comme la macro `\tdoclatexin` s'appuie sur `minted`, on peut utiliser toutes les options prises en compte par `minted`. Voici des exemples d'utilisation.

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

Note.

La macro `\tdoclatexin` est utilisable dans une note de pied de page comme ci-dessous.^a

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

2. Codes tapés directement

Exemple X.3 (Face à face). Afficher un code et son rendu côte à côte se fait comme suit où la macro `\tdoctcb` permet de juste taper `\tdoctcb{sbs}` au lieu de `listing side text` (`sbs` est pour « `s·ide b·y s·ide` », soit « côte à côte » en anglais, tandis que `tcb` est l'abréviation standard de `tcolorbox`). Bien noter l'emploi de chevrons, et non de crochets (nous revenons sur ceci un plus bas).

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

Ceci donne :

$A = B + C$	$A = B + C$
-------------	-------------

Exemple X.4 (À la suite). `\begin{tdoclatex}... \end{tdoclatex}` produit le résultat suivant (ce réglage par défaut s'obtient aussi via l'emploi de `\tdoctcb{std}`).¹³

$A = B + C$

$A = B + C$

Exemple X.5 (Juste le code). Via `\tdoctcb{code}`, on aura juste le code comme ci-après.

$A = B + C$

Exemple X.6 (Personnaliser). L'environnement `tdoclatex` accepte deux types d'argument optionnel.

1. Entre de classiques crochets, on peut employer toute option prise en compte par `minted`.

10. La mise en forme des codes se fait via les packages `minted` et `tcolorbox`.

11. Le nom de la macro `\tdoclatexin` vient de « *in·line* \LaTeX » soit « \LaTeX en ligne » en anglais.

12. Une couleur de fond est volontairement utilisée pour subtilement faire ressortir les codes `\LaTeX`.

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

2. Entre des chevrons, on peut employer toute option prise en compte par les environnements obtenus via `tcolorbox`.

Par exemple, on peut faire les modifications suivantes si besoin.¹⁴

```
\begin{tdoclatex}%
  [linenos, style = igor, showspaces]%
  <\tdoctcb{sbs},
  attach boxed title to top left = {yshift = -9pt},
  fonttitle                       = \bfseries,
  title                           = Modifications locales,
  top                             = 10pt>
% Parfois utile, mais ne pas en abuser !
$A = B + C$
% Fin de cette démonstration.
\end{tdoclatex}
```

Ceci donne :

Modifications locales

<pre>1 <i>%_Parfois_utile,_mais_ne_pas_en_abuser_!</i> 2 <i>\$A = B + C\$</i> 3 <i>%_Fin_de_cette_démonstration.</i></pre>	$A = B + C$
--	-------------

⚠ Avertissement.

Pour obtenir la mise en forme par défaut d'un code commençant par un crochet, ou un chevron, il faudra bidouiller un peu comme ci-dessous.

```
\begin{tdoclatex}[]
[Étrange... Ou pas !]
\end{tdoclatex}
OU.
\begin{tdoclatex}<>
<Étrange... Ou pas !>
\end{tdoclatex}
```

Ceci donne :

```
[Étrange... Ou pas !]
-----
[Étrange... Ou pas !]
```

OU.

```
<Étrange... Ou pas !>
-----
<Étrange... Ou pas !>
```

Une autre méthode consiste à utiliser la primitive `\string` comme ci-après.

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

Ceci donne :

¹⁴ Cette documentation utilise les options entre chevrons pour obtenir des rendus corrects de codes L^AT_EX produisant des cadres ombrés : voir la section 2 page 6.

```
[Étrange... Ou pas !]
```

```
[Étrange... Ou pas !]
```

OU.

```
<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 le même système d'options entre crochets, ou entre chevrons, que l'environnement `tdoclatex`.

Exemple X.7 (Face à face).

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

Ceci produit la mise en forme suivante.

```
% Juste une démo. | xyz = 1  
$x y z = 1$
```

Exemple X.8 (À la suite).

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

Ceci produit la mise en forme suivante qui correspond aussi à l'option `\tdoctcb{std}`.

```
% Juste une démo.  
$x y z = 1$  
-----  
xyz = 1
```

Exemple X.9 (Juste le code).

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

Ceci produit la mise en forme suivante.

```
% Juste une démo.  
$x y z = 1$
```

Exemple X.10 (Personnaliser).

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

Ceci produit la mise en forme suivante.

```
%_Juste_une_démo.  
$x_y_z_=_1$  
-----  
xyz = 1
```

4. Codes importés et mis en situation

Note.

Les textes par défaut de la macro `\tdoclatexshow` tiennent compte de la langue détectée par `tutodoc`.

Exemple X.11 (Mise en situation). `\tdoclatexshow{examples-listing-xyz.tex}` produit ce qui suit.

```
% Juste une démo.  
$x y z = 1$
```

Ceci donne :

$xyz = 1$

Exemple X.12 (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.

```
% Juste une démo.  
$x y z = 1$
```

Voici le rendu réel.

$xyz = 1$

Exemple X.13 (Les options disponibles). En plus du texte explicatif, il est aussi possible d'utiliser toutes les options de l'environnement `tdocshowcase`, voir la section IX page 8. Voici un exemple illustrant ceci.

```
\tdoclatexshow[style      = stripe,  
                  col-stripe = orange,  
                  col-text  = blue!70!black,  
                  before    = Rendu ci-après.,  
                  explain   = Ce qui vient est coloré...,  
                  after     = Rendu fini.,]  
{examples-listing-latex-xyz.tex}
```

Ceci va produire ce qui suit.

```
% Juste une démo.  
$x y z = 1$
```

Ce qui vient est coloré...

————— Rendu ci-après. —————

$xyz = 1$

————— Rendu fini. —————

XI. Présenter du code informatique

Certains packages proposent des fonctionnalités nécessitant de coder un peu en Lua.¹⁵ Pour ces projets, la documentation doit pouvoir présenter des lignes de code Lua ; c'est pour cette raison que `tutodoc` permet de faire cela aisément, et bien plus.¹⁶

Important.

Les outils de cette section permettent aussi de présenter du code \LaTeX , mais il ne faut pas les utiliser pour de simples cas d'utilisation, car les macros et les environnements présentés juste après servent à étudier du code, et non juste à l'employer : se reporter à la section X page 11 pour faire appel aux bons outils pour la mise en forme de cas d'utilisation \LaTeX .

15. Pour les mathématiques, on peut citer `luacas` et `tkz-elements`.

16. La mise en forme des codes étant faite via les packages `minted` et `tcolorbox`, les macros et les environnements présentés dans cette section permettent la mise en forme de codes dans tous les langages supportés par `Pygments`, un projet Python utilisé en coulisse par `minted`.

Exemple XI.2 (Personnalisation ponctuelle du rendu).

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

if name == "" then
    print("Ah, pas très bavard aujourd'hui !")
else
    print("Bonjour " .. name .. ".")
    print("Épatant ! En fait, pas du tout...")
end
\end{tdoccode}
```

Ceci donne :

```
1 io.write("Qui êtes-vous ? ")
2 local name = io.read()
3
4 if name == "" then
5     print("Ah, pas très bavard aujourd'hui !")
6
7 else
8     print("Bonjour " .. name .. ".")
9     print("Épatant ! En fait, pas du tout...")
10 end
```

3. Codes importés

La macro `\tdoccodeinput` attend le langage et le chemin d'un fichier à mettre en forme, et éventuellement d'options similaires à ce que propose l'environnement `tdoccode`.

Exemple XI.3 (Fonctionnement standard).

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

Ceci donne :

```
main :: IO ()
main = do
    putStr "Qui êtes-vous ? "
    name <- getLine

    if name == ""
    then putStrLn "Ah, pas très bavard aujourd'hui !"
    else do
        putStrLn ("Bonjour " ++ name ++ ".")
        putStrLn "Épatant ! En fait, pas du tout..."
```

Exemple XI.4 (Personnalisation ponctuelle du rendu).

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

Ceci donne :

```
1 \NewDocumentCommand{\helloyou}{m}{%
2     \IfBlankTF{#1}{%
3         Ah, pas très bavard aujourd'hui !%
4     }{%
5         Bonjour #1.
6     }
```



```

\paragraph{Un titre bien versionné.}
\tdocversion{1.2.3}[2024-11-23]
Bla, bla...
Ble, ble...

Bli, bli...
Blo, blo...

\tdocdate{2024-11-23}
\paragraph{Un titre mal versionné.}
Blu, blu...

```

Ceci donne :

Un titre bien versionné. *Bla, bla, bla... Ble, ble...*

Bli, bli, bli... Blo, blo...

Un titre mal versionné. *Blu, blu, blu...*

Exemple XII.4 (Régler le positionnement vertical). *Si besoin, on peut modifier le décalage vertical utilisé pour bien placer les dates et les versions dans la marge, la valeur par défaut étant (-8 pt).*

Voilà ce que cela donne sans déplacement vertical.

```

\paragraph{Un réglage maison.}%
\tdocversion{1.2.3}[2024-10-29]<0pt>

Blo, blo...

```

Ceci donne :

Voilà ce que cela donne sans déplacement vertical.

Un réglage maison. *Blo, blo, blo...*

Important.

1. Les macros `\tdocdate` et `\tdocversion` nécessitent deux compilations.
2. Comme la langue détectée 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`.

Mise en garde.

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 \LaTeX 3, 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 changements faits, ou à venir.²⁰

Note.

Concernant les icônes, voir la note au début de la section 2 page 6.

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

a. La sobriété avant tout

Exemple XII.5 (Juste pour la toute première version).

<code>\tdocstartproj{Première version du projet.}</code>	 Première version du projet.
--	---

Exemple XII.6 (Pour les nouveautés).

<code>\begin{tdocnew}</code> <code> \item Info 1...</code> <code> \item Info 2...</code> <code>\end{tdocnew}</code>	 Nouveau. <ul style="list-style-type: none">• Info 1...• Info 2...
--	---

Exemple XII.7 (Pour les mises à jour).

<code>\begin{tdocupdate}</code> <code> \item Info 1...</code> <code> \item Info 2...</code> <code>\end{tdocupdate}</code>	 Mise à jour. <ul style="list-style-type: none">• Info 1...• Info 2...
--	---

Exemple XII.8 (Pour les bifurcations).

<code>\begin{tdocbreak}</code> <code> \item Info 1...</code> <code> \item Info 2...</code> <code>\end{tdocbreak}</code>	 Bifurcation. <ul style="list-style-type: none">• Info 1...• Info 2...
--	---

Exemple XII.9 (Pour les problèmes).

<code>\begin{tdocprob}</code> <code> \item Info 1...</code> <code> \item Info 2...</code> <code>\end{tdocprob}</code>	 Problème. <ul style="list-style-type: none">• Info 1...• Info 2...
--	--

Exemple XII.10 (Pour les réparations).

<code>\begin{tdocfix}</code> <code> \item Info 1...</code> <code> \item Info 2...</code> <code>\end{tdocfix}</code>	 Réparation. <ul style="list-style-type: none">• Info 1...• Info 2...
--	--

Exemple XII.11 (Feuille de route).

<code>\begin{tdoctodo}</code> <code> \item Info 1...</code> <code> \item Info 2...</code> <code>\end{tdoctodo}</code>	 À faire. <ul style="list-style-type: none">• Info 1...• Info 2...
--	---

Exemple XII.12 (Informations techniques).

<code>\begin{tdoctech}</code> <code> \item Info 1...</code> <code> \item Info 2...</code> <code>\end{tdoctech}</code>	 Information technique. <ul style="list-style-type: none">• Info 1...• Info 2...
--	---

Exemple XII.13 (Thématiques aux choix avec une icône).

<code>\begin{tdoctic}{À cacher}<\faEyeSlash></code> <i>% Une icône venant de fontawesome5.</i> <code> \item Info 1...</code> <code> \item Info 2...</code> <code>\end{tdoctic}</code>	 À cacher. <ul style="list-style-type: none">• Info 1...• Info 2...
---	--

Exemple XII.14 (Thématiques aux choix sans icône).

<code>\begin{tdoctic}{La fin des icônes}</code> <code> \item Info 1...</code> <code> \item Info 2...</code> <code>\end{tdoctic}</code>	La fin des icônes. <ul style="list-style-type: none">• Info 1...• Info 2...
---	---

b. De la couleur si besoin

Il peut être utile de mettre en avant certains changements : ceci n'est faisable qu'en modifiant la couleur du contenu.

Exemple XII.15 (Une première version tape-à-l'oeil).

```
\tdocstartproj[DarkOrchid]%  
    {Version 1 chamarrée.}
```

 Version 1 chamarrée.

Exemple XII.16 (Des réparations exceptionnelles).

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

 Réparation.
• Info...

3. Le quoi et le quand

Les clés optionnelles `col-chges`, `date` et `version` permettent de dater et/ou versionner directement un changement d'un type particulier. Voici des exemples d'utilisation.

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

Ceci donne :

29/10/2024  **Information technique.**

- Info...

1.2.3  **Mise à jour.**

- Info...

4.5.6
30/11/2025  **À cacher.**

- Info...

XIII. Décoration

Finissons cette documentation avec un petit outil de mise en forme qui peut rendre de grands services.

<pre>Bla, bla, bla... \tdocsep % Pratique pour délimiter. Ceci fonctionne avec des énumérations. \begin{itemize} \item Point souligné. \end{itemize} \tdocsep % Un comportement uniforme. Ble, ble, ble...</pre>	<p>Bla, bla, bla...</p> <hr/> <p>Ceci fonctionne avec des énumérations.</p> <ul style="list-style-type: none"> • Point souligné. <hr/> <p>Ble, ble, ble...</p>
---	---

XIV. Contribuer

i Note.

Nul besoin d'être un codeur pour participer aux traductions, y compris pour celles utiles au fonctionnement de tutodoc.

1. Compléter les traductions

i Note.

L'auteur de tutodoc gère les versions françaises et anglaises des traductions.

☠ Mise en garde.

Même si nous allons expliquer comment traduire les documentations, il semble peu pertinent de le faire, car l'anglais devrait suffire de nos jours.^a

^a. L'existence d'une version française est une simple conséquence de la langue maternelle de l'auteur de tutodoc.

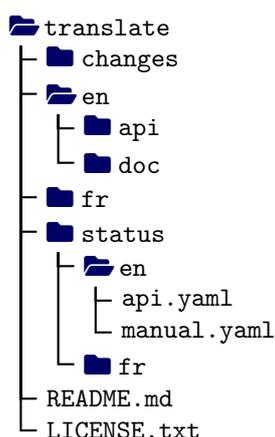


FIGURE 1 – Vue simplifiée du dossier des traductions

Les traductions sont organisées grosso-modo comme dans la figure 1 où seuls les dossiers importants pour les traductions ont été « ouverts ». ²¹ **Un peu plus bas, la section e donne les démarches à suivre pour ajouter de nouvelles traductions.**

21. Cette organisation était celle du 5 octobre 2024.

a. Les dossiers fr et en

Ces deux dossiers, gérés par l'auteur de `tutodoc`, ont la même organisation ; ils contiennent des fichiers faciles à traduire même si l'on n'est pas un codeur.

b. Le dossier changes

Ce dossier est un outil de communication où sont indiqués les changements importants sans s'attarder sur les modifications mineures propres à une ou plusieurs traductions.

c. Le dossier status

Ce dossier permet de savoir où en sont les traductions du point de vue du projet. Tout se passe via des fichiers YAML bien commentés, et lisibles par un non codeur.

d. Les fichiers README.md et LICENSE.txt

Le fichier `LICENSE.txt` est bien nommé, tandis que le fichier `README.md` reprend en anglais les points importants de ce qui est dit dans cette section à propos des nouvelles traductions.

e. De nouvelles traductions

Important.

*Le dossier `api` contient les traductions relatives aux fonctionnalités de `tutodoc`. Vous y trouverez des fichiers de type `TXT` à modifier via un éditeur de texte, ou de code, mais non via un traitement de texte. Les contenus de ces fichiers utilisent des lignes commentées en anglais pour expliquer ce que fera `tutodoc` ; ces lignes commencent par `//`. Voici un extrait de ce type de fichier où **les traductions se font après chaque signe = sans toucher à ce qui se trouve avant**, car ce morceau initial est utilisé en interne par le code de `tutodoc`.*

```
// #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.

Le dossier `doc` est réservé aux documentations. Il contient des fichiers `TEX` compilables directement pour une validation en temps réel des traductions faites.

Avertissement.

Ne partir que de l'un des dossiers `fr` et `en`, car ceux-ci sont de la responsabilité de l'auteur de `tutodoc`.

Imaginons que vous souhaitiez ajouter le support de l'italien en partant de fichiers rédigés en anglais.²²

Méthode 1 : utilisation de `git`.

1. Récupérer le dossier du projet via <https://github.com/bc-tools/for-latex/tree/tutodoc>. Ne pas passer via la branche `main` qui sert à figer les dernières versions stables des projets du dépôt unique <https://github.com/bc-tools/for-latex>.
2. Dans le dossier `tutodoc/contrib/translate`, créer une copie `it` du dossier `en`, le nom court de la langue étant documenté dans la page « *IETF language tag* » de Wikipédia.
3. Une fois la traduction achevée dans le dossier `it`, partager la via <https://github.com/bc-tools/for-latex/tree/tutodoc> en usant d'un classique `git push`.

²². Comme indiqué plus haut, il n'y a pas de besoin réel du côté du dossier `doc`.

Méthode 2 : communiquer par courriels.

1. Via un courriel de sujet « *tutodoc - CONTRIB - en FOR italian* », demander une version des traductions anglaises (noter l'emploi du nom anglais de la nouvelle langue). Bien respecter le sujet du courriel, car l'auteur de *tutodoc* automatise le pré-traitement de ce type de courriels.
2. Vous recevrez un dossier nommé *italian* contenant la version anglaise des dernières traductions. Ce dossier sera le lieu de votre contribution.
3. Une fois la traduction achevée, il faudra compresser votre dossier *italian* au format *zip* ou *rar* avant de l'envoyer par courriel avec le sujet « *tutodoc - CONTRIB - italian* ».

2. Améliorer le code source

Important.

*Si vous souhaitez participer à *tutodoc*, il faudra privilégier le paradigme de programmation \LaTeX 3.*

La participation en tant que codeuse, ou codeur, se fait via le dépôt <https://github.com/bc-tools/for-latex/tree/tutodoc> correspondant à la branche de développement *tutodoc*.

Mise en garde.

*Ne pas passer via la branche *main* qui sert à figer les dernières versions stables des projets du dépôt unique <https://github.com/bc-tools/for-latex>.*

XV. Historique

Réparation.

- Documentation : les références aux outils pour indiquer des changements ont été incorrectement écrites comme caractéristiques des contenus colorés mis en avant.

Bifurcation.

- La macro `\tdocenv` et sa version étoilée ne proposent plus d'option.
- Cas d'utilisation \LaTeX : la présentation par défaut est plus sobre, et des options permettent d'ajouter les lignes cadrantes, ou la bande colorée. Voir juste après.

Nouveau.

- Mise en forme de codes informatiques en plus de ceux spécifiquement en \LaTeX .
 1. Création de `\begin{tdoccode} ... \end{tdoccode}` et de `\tdoccodein`.
 2. Pour les macros pour du code en ligne, et les environnements pour des blocs de code, des options de type `minted` s'indiquent à l'intérieur de crochet de manière traditionnelle : `[options minted]`.
 3. Pour les environnements pour des blocs de code, des options de type `tcolorbox` s'indiquent à l'intérieur de chevrons : `<options tcolorbox>`.
 4. La nouvelle macro `\tdoctcb` permet d'utiliser des raccourcis pour les styles `tcolorbox` régulièrement utilisés.
- Documentation : une nouvelle section présente les outils de mise en forme de codes informatiques autres que les cas d'utilisation de \LaTeX .

Mise à jour.

- Les sous-sous-sections sont numérotées en lettres minuscules.
- Thèmes.
 1. Moins d'espace consommé.
 2. Les ombres ont une meilleure coloration.
 3. Pour tous les thèmes sauf le `draft`, le rayon des arcs des coins des cadres passe de `.75mm` à `.2pt`.
 4. Cas d'utilisation de \LaTeX : avec le thème `color`, la couleur du fond passe de `yellow!4` à `gray!5`.
 5. Dernières modifications : avec le thème `dark`, le texte `[Init]` produit par la macro `\tdocstartproj` utilise la même police que celle des titres des environnements pour indiquer des changements.

🔗 Bifurcation.

- Mise en forme : la classe `scrartcl` remplace la vénérable `article`. Cela implique un meilleur positionnement des notes de marge avec les options retenues pour charger `scrartcl`.
- Code \LaTeX : la macro `\tdocinlatex` a été renommée `\tdoclatexin`.
- Les noms des clés pour les couleurs utiliseront des traits d'union lorsque cela sera nécessaire : cela implique les changements suivants.
 1. Indiquer les derniers changements : l'option `colchges` des environnements a été renommée `col-chges`.
 2. Démonstration de codes \LaTeX : pour l'environnement `tdocshowcase` et la macro `\tdocshowcaseinput`, les options `colstripe` et `coltext` ont été renommées `col-stripe` et `col-text`.

🔧 Réparation.

- Mise en avant colorée de contenus : pour les `\newkeytheorem` utilisés avec le thème `draft`, il a fallu ajouter `postheadhook = \leavevmode` (ceci est nécessaire car le contenu peut juste être de type liste).

💎 Nouveau.

- Documentation : ajout d'une section listant les dépendances.
- Options de classe.
 1. Les options qui ne sont pas spécifiques à `tutodoc` sont transmises à la classe chargée de la mise en forme générale.
 2. Les options `fontsize` et `DIV` de la classe `scrartcl` ne peuvent pas être utilisées, car leurs valeurs sont fixées par `tutodoc`.
- La macro `\tdocinEN` respecte les règles linguistiques anglaises.
- Indiquer les derniers changements.
 1. Ajout de l'environnement `\begin{tdoctodo} ... \end{tdoctodo}`.
 2. Chaque environnement dispose d'une nouvelle option `col` pour la couleur du contenu indiquant les changements.

🔄 Mise à jour.

- Le thème `draft` et les changements : les environnements pour les derniers changements n'emploient plus d'icone.
- Documentation : la galerie des thèmes utilise un meilleur exemple factice.

🔧 Information technique.

- Organisation simplifiée des fichiers de configuration dans le projet final.
 1. Emploi d'un fichier par thème avec des noms du type `tutodoc-*.css.cls`.
 2. Localisation : utilisation de noms comme `tutodoc-*.loc.cls`.
-

💎 Nouveau.

- Les macros `\tdocdate` et `\tdocversion` ont un nouvel argument final facultatif `<voffset>` pour modifier le décalage vertical.
- De meilleurs environnements pour indiquer les modifications apportées.
 1. Les nouvelles clés facultatives `col`, `date` et `version` permettent d'indiquer la date et la version d'une modification apportée à un sujet spécifique.
 2. Utilisation de `\paragraph` pour le titre.

🔄 Mise à jour.

- Changements : la police des notes de marge aura toujours une forme normale.
 - Ornement : utilisation d'un `\cleaders` pour éviter les lignes orphelines en bas de page.
-

🔧 Information technique.

- Les règles de nommage de `CTAN` nécessitent l'usage de noms du type `tutodoc-*.css.cls.sty` pour les fichiers à-la CSS.
-

🔗 Bifurcation.

- L'environnement `showcase` et ses descendants : la clé `color` a été renommée `colstripe`.
- La macro `\tdoclinkcolor` devient la couleur `tutodoc@link@color` destinée à un usage interne.

◆ Nouveau.

- L'option de classe `theme` permet de choisir différents thèmes de mise en forme.
- Journal des modifications : ajout de l'environnement `tdoctech` pour les informations techniques.
- L'environnement `showcase` et ses descendants : la clé `coltext` permet de changer aussi la couleur du texte.
- Les nouvelles fonctionnalités ont été documentées.

🔄 Mise à jour.

- Journal des modifications : l'environnement `tdocupdate` utilise l'icône  au lieu de .

🔧 Réparation.

- Les traductions espagnoles n'avaient pas été livrées dans la précédente version ! Ne pas rire trop fort...
-

🔧 Information technique.

- La version 3 de `minted` est prise en compte.

🔗 Bifurcation.

- La classe `tutodoc` remplace le défunt package `tutodoc` (pour le moment, la toute jeune classe ne propose aucune option spécifique).
- La macro `\tdocruler` s'emploie via `\tdocruler[<color>]{<text>}` au lieu de `\tdocruler{<text>}{<color>}`.

◆ Nouveau.

- La classe est utilisable en langue espagnole.
- La documentation contient une nouvelle section expliquant comment contribuer.

🔧 Réparation.

- La macro `\tdocdate` ne gérait pas le format et la mise en forme de la date.
 - Les cadres colorés ne coloraient pas le texte après un saut de page.
-

🔗 Bifurcation.

- L'environnement `tdoccaution` a été renommé `tdoccaut` pour une saisie simplifiée.
- Mise en avant de contenus : les exemples et remarques, indiqués via les environnements `tdocexa` et `tdocrem`, sont numérotés via un compteur commun.
- La macro inutilisée `\tdocxspace` a été supprimée.

◆ Nouveau.

- Journal des changements : la macro `\tdocstartproj` permet de gérer le cas de la première version publique.
- Factorisation du code : la macro `\tdocicon` se charge de l'ajout d'icônes devant du texte.

🔄 Mise à jour.

- Couleurs : les macros `\tdocdarkcolor` et `\tdoclightcolor` proposent un argument facultatif.
 1. `\tdocdarkcolor` : la quantité de couleur par rapport au noir peut être définie de manière facultative.
 2. `\tdoclightcolor` : le taux de transparence peut être défini de manière facultative.
 - Mise en avant de contenus : réduction de l'espace autour du contenu dans les cadres colorés.
 - Gestion des versions : un meilleur espacement vertical via `\vphantom`.
-

◆ Nouveau.

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

🔧 Information technique.

- 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 temporairement l'usage de la version 2 de `minted`.

🔗 Bifurcation.

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

1.5.0
19/10/2024

1.4.0
28/09/2024

1.3.1
26/09/2024

1.3.0
25/09/2024

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`
-

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.
-

Nouveau.

- Journal des changements : deux nouveaux environnements.
 1. `\begin{tdocbreak} ... \end{tdocbreak}` pour les « *bifurcations* », c'est-à-dire les modifications non rétrocompatibles.
 2. `\begin{tdocprob} ... \end{tdocprob}` pour les problèmes repérés.
 - `\tdoclatexin` : un jaune léger est utilisé comme couleur de fond.
-

Réparation.

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

Première version publique du projet.

1.2.0-a
23/08/2024

1.1.0
06/01/2024

1.0.1
08/12/2023

1.0.0
29/11/2023

Annexe – Galerie des thèmes

i Note.

Chaque exemple est un PDF directement inséré dans ce document (ne soyez donc pas surpris par la numérotation des pages).

Le thème "bW"

I. Liens

Un lien très gros, mais au moins on le voit.

II. Des codes L^AT_EX

Taper du code L^AT_EX en ligne comme `E = m c^2 \neq \pi \neq \frac{3}{14}` est utile, tout comme montrer des cas d'utilisation comme le suivant.

Du code `\LaTeX` mis en forme, c'est top : $E = m c^2$ ou $\pi \neq \frac{3}{14}$.

Du code L^AT_EX mis en forme, c'est top : $E = mc^2$ ou $\pi \neq \frac{3}{14}$.

On dispose aussi d'un mode côte-à-côte moins envahissant. Sympa! Non?

Du code <code>\LaTeX</code> mis en forme, c'est top: $E = m c^2$ ou $\pi \neq \frac{3}{14}$.	Du code L ^A T _E X mis en forme, c'est top : $E = mc^2$ ou $\pi \neq \frac{3}{14}$.
---	---

III. Mettre en avant, versionner et dater

1. tdocexa, tdocrem

Dans le flot du texte, il est toujours utile de pouvoir indiquer des exemples et des remarques qui viennent compléter le contenu principal.

Exemple III.1. *Que dire¹? Je ne sais pas, mais c'est sympathique. Non ?*

Remarque III.2. *Que dire²? Je ne sais pas, mais c'est sympathique. Non ?*

2. tdocnote, tdoctip...

Suivant le contexte d'utilisation, il est parfois nécessaire de pouvoir mettre en avant des contenus en indiquant leur degré d'importance.

Note.

Que dire^a? Je ne sais pas, mais c'est sympathique. Non ?

a. N'oublions pas les notes de bas de page...

Astuce.

Que dire ? Je ne sais pas, mais c'est sympathique. Non ?

Important.

Que dire ? Je ne sais pas, mais c'est sympathique. Non ?

Mise en garde.

Que dire ? Je ne sais pas, mais c'est sympathique. Non ?

Avertissement.

Que dire ? Je ne sais pas, mais c'est sympathique. Non ?

1. N'oublions pas les notes de bas de page...
2. N'oublions pas les notes de bas de page...

3. tdocbreak, tdocfix...

 Une nouvelle section démonstrative...

À faire.

- Une galerie serait bienvenue...

Dans un journal de bord, il est important de bien visualiser les types de changements. Ceci rend plus efficace la lecture côté utilisateur !

Bifurcation.

- Infos...

Information technique.

- Infos...

Réparation.

- Infos...

Mise à jour.

- Infos...

Nouveau.

- Infos...

À faire.

- Infos...

Problème.

- Infos...

Le thème "color"

I. Liens

Un lien très gros, mais au moins on le voit.

II. Des codes L^AT_EX

Taper du code L^AT_EX en ligne comme `E = m c^2 \neq \pi \neq \frac{3}{14}` est utile, tout comme montrer des cas d'utilisation comme le suivant.

Du code `\LaTeX` mis en forme, c'est top : `$E = m c^2$` ou `$$\pi \neq \frac{3}{14}$$`.

Du code L^AT_EX mis en forme, c'est top : `E = mc^2` ou `\pi \neq \frac{3}{14}`.

On dispose aussi d'un mode côte-à-côte moins envahissant. Sympa! Non ?

Du code <code>\LaTeX</code> mis en forme, c'est top: <code>\\ \$E = m c^2\$</code> ou <code>\$\$\pi \neq \frac{3}{14}\$\$</code> .	Du code L ^A T _E X mis en forme, c'est top : <code>E = mc^2</code> ou <code>\pi \neq \frac{3}{14}</code> .
--	---

III. Mettre en avant, versionner et dater

1. tdocexa, tdocrem

Dans le flot du texte, il est toujours utile de pouvoir indiquer des exemples et des remarques qui viennent compléter le contenu principal.

Exemple III.1. *Que dire¹? Je ne sais pas, mais c'est sympathique. Non ?*

Remarque III.2. *Que dire²? Je ne sais pas, mais c'est sympathique. Non ?*

2. tdocnote, tdoctip...

Suivant le contexte d'utilisation, il est parfois nécessaire de pouvoir mettre en avant des contenus en indiquant leur degré d'importance.

Note.

Que dire^a? Je ne sais pas, mais c'est sympathique. Non ?

a. N'oublions pas les notes de bas de page...

Astuce.

Que dire ? Je ne sais pas, mais c'est sympathique. Non ?

Important.

Que dire ? Je ne sais pas, mais c'est sympathique. Non ?

Mise en garde.

Que dire ? Je ne sais pas, mais c'est sympathique. Non ?

Avertissement.

Que dire ? Je ne sais pas, mais c'est sympathique. Non ?

1. N'oublions pas les notes de bas de page...

2. N'oublions pas les notes de bas de page...

3. tdocbreak, tdocfix...

 Une nouvelle section démonstrative...

À faire.

- Une galerie serait bienvenue...

Dans un journal de bord, il est important de bien visualiser les types de changements. Ceci rend plus efficace la lecture côté utilisateur !

Bifurcation.

- Infos...

Information technique.

- Infos...

Réparation.

- Infos...

Mise à jour.

- Infos...

Nouveau.

- Infos...

À faire.

- Infos...

Problème.

- Infos...

Le thème "dark"

I. Liens

Un lien très gros, mais au moins on le voit.

II. Des codes L^AT_EX

Taper du code L^AT_EX en ligne comme `E = m c^2 \neq \pi \neq \frac{3}{14}` est utile, tout comme montrer des cas d'utilisation comme le suivant.

Du code `\LaTeX` mis en forme, c'est top : `$E = m c^2$` ou `$$\pi \neq \frac{3}{14}$$`.

Du code L^AT_EX mis en forme, c'est top : $E = mc^2$ ou $\pi \neq \frac{3}{14}$.

On dispose aussi d'un mode côte-à-côte moins envahissant. Sympa! Non?

Du code <code>\LaTeX</code> mis en forme, c'est top: <code>\\</code> <code>\$E = m c^2\$</code> ou <code>\$\$\pi \neq \frac{3}{14}\$\$</code> .	Du code L ^A T _E X mis en forme, c'est top : $E = mc^2$ ou $\pi \neq \frac{3}{14}$.
--	--

III. Mettre en avant, versionner et dater

1. tdocexa, tdocrem

Dans le flot du texte, il est toujours utile de pouvoir indiquer des exemples et des remarques qui viennent compléter le contenu principal.

Exemple III.1. *Que dire¹? Je ne sais pas, mais c'est sympathique. Non ?*

Remarque III.2. *Que dire²? Je ne sais pas, mais c'est sympathique. Non ?*

2. tdocnote, tdoctip...

Suivant le contexte d'utilisation, il est parfois nécessaire de pouvoir mettre en avant des contenus en indiquant leur degré d'importance.

Note.

Que dire ? Je ne sais pas, mais c'est sympathique. Non ?

a. N'oublions pas les notes de bas de page...

Astuce.

Que dire ? Je ne sais pas, mais c'est sympathique. Non ?

Important.

Que dire ? Je ne sais pas, mais c'est sympathique. Non ?

Mise en garde.

Que dire ? Je ne sais pas, mais c'est sympathique. Non ?

Avertissement.

Que dire ? Je ne sais pas, mais c'est sympathique. Non ?

1. N'oublions pas les notes de bas de page...

2. N'oublions pas les notes de bas de page...

3. tdocbreak, tdocfix...

 Une nouvelle section démonstrative...

 **À faire.**

- Une galerie serait bienvenue...

Dans un journal de bord, il est important de bien visualiser les types de changements. Ceci rend plus efficace la lecture côté utilisateur !

 **Bifurcation.**

- Infos...

 **Réparation.**

- Infos...

 **Nouveau.**

- Infos...

 **Problème.**

- Infos...

 **Information technique.**

- Infos...

 **Mise à jour.**

- Infos...

 **À faire.**

- Infos...

Le thème "draft"

I. Liens

Un lien très gros, mais au moins on le voit.

II. Des codes \LaTeX

Taper du code \LaTeX en ligne comme $E = m c^2 \neq \pi \neq \frac{3}{14}$ est utile, tout comme montrer des cas d'utilisation comme le suivant.

Du code \LaTeX mis en forme, c'est top : $E = m c^2$ ou $\pi \neq \frac{3}{14}$.

Du code \LaTeX mis en forme, c'est top : $E = mc^2$ ou $\pi \neq \frac{3}{14}$.

On dispose aussi d'un mode côte-à-côte moins envahissant. Sympa! Non?

Du code \LaTeX mis en forme, c'est top : $E = m c^2$ ou $\pi \neq \frac{3}{14}$. Du code \LaTeX mis en forme, c'est top : $E = mc^2$ ou $\pi \neq \frac{3}{14}$.

III. Mettre en avant, versionner et dater

1. tdocexa, tdocrem

Dans le flot du texte, il est toujours utile de pouvoir indiquer des exemples et des remarques qui viennent compléter le contenu principal.

Exemple III.1. *Que dire¹? Je ne sais pas, mais c'est sympathique. Non ?*

Remarque III.2. *Que dire²? Je ne sais pas, mais c'est sympathique. Non ?*

2. tdocnote, tdoctip...

Suivant le contexte d'utilisation, il est parfois nécessaire de pouvoir mettre en avant des contenus en indiquant leur degré d'importance.

Note III.3. *Que dire³? Je ne sais pas, mais c'est sympathique. Non ?*

Astuce III.4. *Que dire ? Je ne sais pas, mais c'est sympathique. Non ?*

Important III.5. *Que dire ? Je ne sais pas, mais c'est sympathique. Non ?*

Mise en garde III.6. *Que dire ? Je ne sais pas, mais c'est sympathique. Non ?*

Avertissement III.7. *Que dire ? Je ne sais pas, mais c'est sympathique. Non ?*

3. tdocbreak, tdocfix...

^[Unit] Une nouvelle section démonstrative...

À faire.

- Une galerie serait bienvenue...

Dans un journal de bord, il est important de bien visualiser les types de changements. Ceci rend plus efficace la lecture côté utilisateur !

Bifurcation.

- Infos...

Réparation.

- Infos...

Nouveau.

- Infos...

Problème.

- Infos...

Information technique.

- Infos...

Mise à jour.

- Infos...

À faire.

- Infos...

1. N'oublions pas les notes de bas de page...
2. N'oublions pas les notes de bas de page...
3. N'oublions pas les notes de bas de page...