

# robust-externalize

Tout mettre en cache, et bien plus encore

Léo COLISSON PALAIS

[leo.colisson-palais@univ-grenoble-alpes.fr](mailto:leo.colisson-palais@univ-grenoble-alpes.fr)

## {robust-externalize}

- Projet commencé en mars 2023
- <https://github.com/leo-colisson/robust-externalize>
- $\approx 4700$  lignes de code (hors documentation)
- +  $\approx 120$  pages de documentation
- Publié sur CTAN
- Je suis le créateur et mainteneur du paquet

# Problème initial

# Problème initial : TikZ (et L<sup>A</sup>T<sub>E</sub>X) est lent



Temps de compilation de ma thèse  $\approx 20\text{mn}$  par itération  
⇒ **Total  $\geq 1\text{h}$  de compilation!!**  
(impossible à débugger...)

# Problème initial : TikZ (et L<sup>A</sup>T<sub>E</sub>X) est lent



Aussi un problème sur  
+ petits article !  
(e.g. 5mn déjà long...)

Temps de compilation de ma thèse  $\approx 20\text{mn}$  par itération

$\Rightarrow \text{Total} \geq 1\text{h de compilation!!}$

(impossible à débugger...)

# Solution: librairie externalize?

## 52 Externalization Library

by Christian Feuersänger

### TikZ Library `external`

```
\usetikzlibrary{external} % LATEX and plain TEX  
\usetikzlibrary[external] % ConTeXt
```

This library provides a high-level automatic or semi-automatic export feature for *TikZ* pictures. Its purpose is to convert each picture to a separate PDF without changing the document as such.

It also externalizes `\label` information (and other aux file related stuff) using auxiliary files.

# externalize inutilisable : i) rajoute image = recompile tout

external library rebuilds figures when figure order changes #758

 Open



mkuron opened on Oct 24, 2019

Contributor ...

Assig

No o

Label

Fea

Type

No ty

Proje

No pi

Miles

...

N

Relat

When using `\usetikzlibrary{external}`, the externalized pictures get written to files named like `<main file name>-figure<number>`. As I just discussed with [@hmenke](#), it would be useful if files could optionally be named by the MD5 sum (which is already calculated to detect changes). That would eliminate the need to rebuild unchanged figures, e.g. when reverting to a previous version of a figure, when changing figure order, or when inserting a new figure between existing ones. I am aware that this can cause the number of files to get quite large as old ones are never deleted/overwritten, so the default behavior should not be changed.

  5  1



hmenke on Dec 15, 2019

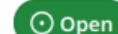
Member ...

This is actually harder than I had anticipated, because the code of the externalization library is mostly unreadable gibberish.

  3

# externalize inutilisable : i) rajoute image = recompile tout

external library rebuilds figures when figure order changes #758

 Open



mkuron opened on Oct 24, 2019

Contributor ...

Assig

No o

Label

Fea

Type

No ty

Proje

No pi

Miles

...

N

Relat

When using `\use{tikzlibrary}{external}`, the externalized pictures get written to files named like `<main file name>-figure<number>`. As I just discussed with [@hmenke](#), it would be useful if files could optionally be named by the MD5 sum (which is already calculated to detect changes). That would eliminate the need to rebuild unchanged figures, e.g. when reverting to a previous version of a figure, when changing figure order, or when inserting a new figure between existing ones. I am aware that this can cause the number of files to get quite large as old ones are never deleted/overwritten, so the default behavior should not be changed.

  5  1



hmenke on Dec 15, 2019

Member ...

This is actually harder than I had anticipated, because the code of the externalization library is mostly unreadable gibberish.

  3

# externalize inutilisable : i) rajoute image = recompile tout

external library rebuilds figures when figure order changes #758

Open



mkuron opened on Oct 24, 2019

Contributor ...

Assig

No o

Label

Fea

Type

No ty

Proje

No pi

Miles

...

N

Relat

When using `\usetikzlibrary{external}`, the externalized pictures get written to files named like `<main file name>-figure<number>`. As I just discussed with [@hmenke](#), it would be useful if files could optionally be named by the MD5 sum (which is already calculated to detect changes). That would eliminate the need to rebuild unchanged figures, e.g. when reverting to a previous version of a figure, when changing figure order, or when inserting a new figure between existing ones. I am aware that this can cause the number of files to get quite large as old ones are never deleted/overwritten, so the default behavior should not be changed.

1 5 1



hmenke on Dec 15, 2019

Member ...

This is actually harder than I had anticipated, because the code of the externalization library is mostly unreadable gibberish.

3

# externalize inutilisable : i) rajoute image = recompile tout

external library rebuilds figures when figure order changes #758

Open



mkuron opened on Oct 24, 2019

Contributor ...

Assig

No o

Label

Fee

Type

No ty

Proje

No pi

Miles

0

N

Relat

When using `\usetikzlibrary{external}`, the externalized pictures get written to files named like `<main file name>-figure<number>`. As I just discussed with [@hmenke](#), it would be useful if files could optionally be named by the MD5 sum (which is already calculated to detect changes). That would eliminate the need to rebuild unchanged figures, e.g. when reverting to a previous version of a figure, when changing figure order, or when inserting a new figure between existing ones. I am aware that this can cause the number of files to get quite large as old ones are never deleted/overwritten, so the default behavior should not be changed.



→ our solution  
= hash the content



hmenke on Dec 15, 2019

Member ...

This is actually harder than I had anticipated, because the code of the externalization library is mostly unreadable gibberish.



# externalize inutilisable : ii) code = “unreadable gibberish”

external library rebuilds figures when figure order changes #758

Open



mkuron opened on Oct 24, 2019

Contributor ...

Assig

No o

Label

Fea

Type

No ty

Proje

No pi

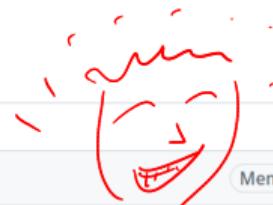
Miles

0

N

Relat

When using `\usetikzlibrary{external}`, the externalized pictures get written to files named like `<main file name>-figure<number>`. As I just discussed with @hmenke, it would be useful if files could optionally be named by the MD5 sum (which is already calculated to detect changes). That would eliminate the need to rebuild unchanged figures, e.g. when reverting to a previous version of a figure, when changing figure order, or when inserting a new figure between existing ones. I am aware that this can cause the number of files to get quite large as old ones are never deleted/overwritten, so the default behavior should not be changed.



hmenke on Dec 15, 2019

Member ...

This is actually harder than I had anticipated, because the code of the externalization library is mostly unreadable gibberish.



## externalize inutilisable: *iii)* preambule du document

Comment (TikZ) externalize fonctionne (simplifié) :

Pour chaque image :

- copier le préambule du document + l'image actuelle dans un nouveau fichier
- compile ce fichier
- insère le PDF

## externalize inutilisable: *iii) preamble du document*

Comment (TikZ) externalize fonctionne (simplifié) :

Pour chaque image :

- copier le préambule du document + l'image actuelle dans un nouveau fichier
- compile ce fichier
- insère le PDF

Si le document charge beaucoup de paquets = **1<sup>re</sup> compilation très longue**:

- 10s pour charger les paquets
- × 200 images à 1s de compilation par image
- 33mn avec externalize, 3mn sans externalize

## { robust-externalize }

Notre **solution** :

- **préambule minimal** (possiblement différent) pour chaque image
- semi **auto-détection** des paquets à charger
- la compilation **en parallèle** des images est possible

## externalize inutilisable: iv) impure

(TikZ) `externalize` est **impure** : changer un macro = pas de recompilation !

## externalize inutilisable: iv) impure

(TikZ) externalize est **impure** : changer un macro = pas de recompilation !

```
\def\foo{My macro}
\begin{tikzpicture}
  \node[fill=green!50!white, rounded corners]{\foo};
\end{tikzpicture}
```

## externalize inutilisable: iv) impure

(TikZ) externalize est **impure** : changer un macro = pas de recompilation !

```
\def\foo{My macro}
\begin{tikzpicture}
  \node[fill=green!50!white, rounded corners]{\foo};
\end{tikzpicture}
```

⇒ My macro

## externalize inutilisable: iv) impure

(TikZ) externalize est **impure** : changer un macro = pas de recompilation !

```
\def\foo{My updated macro}
\begin{tikzpicture}
  \node[fill=green!50!white, rounded corners]{\foo};
\end{tikzpicture}
```

## externalize inutilisable: iv) impure

(TikZ) externalize est **impure** : changer un macro = pas de recompilation !

```
\def\foo{My updated macro}  
\begin{tikzpicture}  
  \node[fill=green!50!white, rounded corners]{\foo};  
\end{tikzpicture}
```

⇒ My macro



## { robust-externalize }

Notre **solution** = **contrôle fin de la pureté**

- on peut choisir quels macros doivent forcer une recompilation et quels macros sont impures
  - système de transfer de macros/couleurs/...
  - auto-détection des macros utilisés
    - ↪ partie **importante et difficile** du projet (but = rester efficace et pure)
- ⇒ cf exemples plus tard

## But = pureté absolue



Recompiler **tout** ce qui doit être recompilé,  
mais **sans recompilations inutiles**

## externalize inutilisable: iv) spécifique à Tikz

Avec (Tikz) externalize, impossible de mettre en cache autre chose  
**que des images Tikz**

## {robust-externalize}

Notre **solution** = **système générique** de mise en cache. Il suffit de spécifier:

- un template de fichier (code source)
- une commande de compilation
- une commande d'inclusion du résultat dans le document principal
- (optionnel) des fichiers de dépendance (changement = recompilation)

## Exemple : includegraphics avec URL

```
\robExtConfigure{
    new preset={my online img} {
        set template={} , % no code needed
        set compilation command= {
            wget "__ROBEXT_MAIN_CONTENT__" -O "__ROBEXT_OUTPUT_PDF__" },
        custom include command=%
            \evalPlaceholder{%
                \includegraphics[__MY_OPTIONS__]{
                    \robExtAddCachePath{__ROBEXT_OUTPUT_PDF__}}}}}
}
\NewDocumentCommand{\includegraphicsOnline}{D<>{}0{}m}{%
    \robExtCacheMe[my online img,
    set placeholder=__MY_OPTIONS__]{#2},#1}{#3}
}
\includegraphicsOnline[width=3cm]{https://placebear.com/200/300}
```

## Exemple : includegraphics avec URL

```
\robExtConfigure{
    new preset={my online img} {
        set template={} , % no code needed
        set compilation command= {
            wget "__ROBEXT_MAIN_CONTENT__" -O "__ROBEXT_OUTPUT_PDF__" },
        custom include command=%
            \evalPlaceholder%
                \includegraphics[__MY_OPTIONS__]{
                    \robExtAddCachePath{__ROBEXT_OUTPUT_PDF__}}}}}
}

\NewDocumentCommand{\includegraphicsOnline}{D<>{}0{}m}{%
    \robExtCacheMe[my online img,
    set placeholder=__MY_OPTIONS__]{#2},#1}{#3}
}

\includegraphicsOnline[width=3cm]{https://placebear.com/200/300}
```

## Exemple : includegraphics avec URL

```
\robExtConfigure{
    new preset={my online img} {
        set template={}, % no code needed
        set compilation command={
            wget "__ROBEXT_MAIN_CONTENT__" -O "__ROBEXT_OUTPUT_PDF__"}, 
        custom include command=%
            \evalPlaceholder{%
                \includegraphics[__MY_OPTIONS__]{
                    \robExtAddCachePath{__ROBEXT_OUTPUT_PDF__}}}}}
}
\NewDocumentCommand{\includegraphicsOnline}{D<>{}0{}m}{%
    \robExtCacheMe[my online img,
    set placeholder=__MY_OPTIONS__]{#2},#1}{#3}
}
\includegraphicsOnline[width=3cm]{https://placebear.com/200/300}
```

## Exemple : includegraphics avec URL

```
\robExtConfigure{
    new preset={my online img} {
        set template={}, % no code needed
        set compilation command={
            wget "__ROBEXT_MAIN_CONTENT__" -O "__ROBEXT_OUTPUT_PDF__"}, 
        custom include command=%
            \evalPlaceholder{%
                \includegraphics[__MY_OPTIONS__]{
                    \robExtAddCachePath{__ROBEXT_OUTPUT_PDF__}}}}}
}
\NewDocumentCommand{\includegraphicsOnline}{D<>{}0{}m}{%
    \robExtCacheMe[my online img,
    set placeholder=__MY_OPTIONS__}{#2},#1}{#3}
}
\includegraphicsOnline[width=3cm]{https://placebear.com/200/300}
```

## Exemple : includegraphics avec URL

```
\robExtConfigure{
    new preset={my online img} {
        set template={} , % no code needed
        set compilation command= {
            wget "__ROBEXT_MAIN_CONTENT__" -O "__ROBEXT_OUTPUT_PDF__" },
        custom include command=%
            \evalPlaceholder%
                \includegraphics[__MY_OPTIONS__]{
                    \robExtAddCachePath{__ROBEXT_OUTPUT_PDF__}}}}}
}
\NewDocumentCommand{\includegraphicsOnline}{D<>{}0{}m}{%
    \robExtCacheMe[my online img,
    set placeholder=__MY_OPTIONS__}{#2},#1}{#3}
}
\includegraphicsOnline[width=3cm]{https://placebear.com/200/300}
```

## Exemple : includegraphics avec URL

```
\robExtConfigure{
    new preset={my online img} {
        set template={}, % no code needed
        set compilation command={
            wget "__ROBEXT_MAIN_CONTENT__" -O "__ROBEXT_OUTPUT_PDF__"}, 
        custom include command=%
            \evalPlaceholder%
                \includegraphics[__MY_OPTIONS__]{
                    \robExtAddCachePath{__ROBEXT_OUTPUT_PDF__}}}}
}
\NewDocumentCommand{\includegraphicsOnline}{D<>{}0{}m}{%
    \robExtCacheMe[my online img,
    set placeholder=__MY_OPTIONS__}{#2}, {#1}{#3}
}
\includegraphicsOnline[width=3cm]{https://placebear.com/200/300}
```

placeholders

## Exemple : includegraphics avec URL

```
\robExtConfigure{
    new preset={my online img} {
        set template={}, % no code needed
        set compilation command={
            wget "__ROBEXT_MAIN_CONTENT__" -O "__ROBEXT_OUTPUT_PDF__" },
        custom include command=%
            \evalPlaceholder{%
                \replacePlaceholdersWithTheirValue
                \includegraphics[__MY_OPTIONS__]{
                    \robExtAddCachePath{__ROBEXT_OUTPUT_PDF__}}}}}
\NewDocumentCommand{\includegraphicsOnline}{D<>{}0{}m}{%
    \robExtCacheMe[my online img,
    set placeholder=__MY_OPTIONS__]{#2},#1}{#3}
}
\includegraphicsOnline[width=3cm]{https://placebear.com/200/300}
```

## Exemple : includegraphics avec URL

```
\robExtConfigure{
    new preset={my online img} {
        set template={}, % no code needed
        set compilation command={
            wget "__ROBEXT_MAIN_CONTENT__" -O "__ROBEXT_OUTPUT_PDF__" ,
        custom include command=%
            \evalPlaceholder{%
                \includegraphics[MY_OPTIONS]{
                    \robExtAddCachePath{__ROBEXT_OUTPUT_PDF__}}}}}
    }
    \NewDocumentCommand{\includegraphicsOnline}{D<>{}0{}m}{%
        \robExtCacheMe[my online img,
        set placeholder=__MY_OPTIONS__]{#2},#1}{#3}
    }
\includegraphicsOnline[width=3cm]{https://placebear.com/200/300}
```

## Exemple : includegraphics avec URL

```
\robExtConfigure{
    new preset={my online img} {
        set template={}, % no code needed
        set compilation command={
            wget "__ROBEXT_MAIN_CONTENT__" -O "__ROBEXT_OUTPUT_PDF__" ,
        custom include command=%
            \evalPlaceholder{%
                \includegraphics[MY_OPTIONS]{
                    \robExtAddCachePath{__ROBEXT_OUTPUT_PDF__}}}}}
    }  
    replace  
    place  
    holders  
    with their value  

    }  

\NewDocumentCommand{\includegraphicsOnline}{D<>{}0{}{}m}{%
    \robExtCacheMe[my online img,
    set placeholder={MY_OPTIONS}{#2},#1]{#3}
}
\includegraphicsOnline[width=3cm]{https://placebear.com/200/300}
```

Annotations on the code:

- A blue bracket labeled "placeholders" covers the line "set placeholder={MY\_OPTIONS}{#2},#1]{#3}".
- A green bracket labeled "wrapper (easier to use)" covers the entire definition of the command \NewDocumentCommand{\includegraphicsOnline}{...}.

## Exemple : includegraphics avec URL

```
\robExtConfigure{
    new preset={my online img} {
        set template={}, % no code needed
        set compilation command={
            wget "__ROBEXT_MAIN_CONTENT__" -O "__ROBEXT_OUTPUT_PDF__" ,
        custom include command=%
            \evalPlaceholder{%
                \includegraphics[MY_OPTIONS]{
                    \robExtAddCachePath{__ROBEXT_OUTPUT_PDF__}}}}}
    }  
replace  
place  
holders  
with their value  

    }  

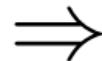
\NewDocumentCommand{\includegraphicsOnline}{D<>{}0{}{}m}{%
    \robExtCacheMe[my online img,
    set placeholder=__MY_OPTIONS__]{#2},#1}{#3}
}
\includegraphicsOnline[width=3cm]{https://placebear.com/200/300}
```

Annotations:

- placeholders**: A blue bracket groups the two occurrences of `__ROBEXT_OUTPUT_PDF__`.
- output path**: A blue bracket groups the output file path `__ROBEXT_OUTPUT_PDF__`. A note below it says `(.pdf but may not be pdf)`.
- wrapper**: A bracket groups the entire command definition `\NewDocumentCommand{\includegraphicsOnline}{D<>{}0{}{}m}{%` ... `}` `}`.
- easier to use**: A note below the `wrapper` bracket says `(easier to use)`.

## Exemple : `includegraphics` avec URL

```
\includegraphics[width=3cm]{https://placebear.com/200/300}
```



# Application : images par IA

**Exercice** : Adapter le précédent exemple pour générer avec l'IA Gemini des images en tapant :

\includegraphics[width=3cm]{A spy cat looking like James-Bond}

en se basant sur l'API :



```
curl -s -X POST \
  "https://generativelanguage.googleapis.com/v1beta/models/gemini-2.5-flash-image:generateContent" \
  -H "x-goog-api-key: $GEMINI_API_KEY" \
  -H "Content-Type: application/json" \
  -d '{
    "contents": [
      "parts": [
        {"text": "Create a picture of a nano banana dish in a fancy restaurant with a Gemini theme"}
      ]
    }
  }' | grep -o '"data": "[^"]*"' | cut -d '"' -f4 \
| base64 --decode > gemini-native-image.png
```

# Aperçu des fonctionnalités de robust-externalize

# Aperçu des fonctionnalités

- Gestion des images My node respecting baseline ❤. avec overlay et depth.

I am an overlay text

# Aperçu des fonctionnalités

- Gestion des images My node respecting baseline ❤️, avec overlay et depth.
- Détection semi-automatique des paquets/macros à charger.

I am an overlay text

# Aperçu des fonctionnalités

- Gestion des images My node respecting baseline ❤️, avec overlay et depth.
- Détection semi-automatique des paquets/macros à charger.
- Gestion fine de la pureté (quand recompiler automatiquement).

I am an overlay text

# Aperçu des fonctionnalités

- Gestion des images My node respecting baseline ❤️, avec overlay et depth.
- Détection semi-automatique des paquets/macros à charger.
- Gestion fine de la pureté (quand recompiler automatiquement).
- Automatiquement mettre en cache certaines commandes et environnements via:  
`\cacheEnvironment{forest}{\usepackage{forest}}`

I am an overlay text

# Aperçu des fonctionnalités

- Gestion des images My node respecting baseline ❤️, avec overlay et depth.
- Détection semi-automatique des paquets/macros à charger.
- Gestion fine de la pureté (quand recompiler automatiquement).
- Automatiquement mettre en cache certaines commandes et environnements via:  
`\cacheEnvironment{forest}{latex, add to preamble={\usepackage{forest}}}`
- Possibilité de désactiver le cache pour certaines commandes/images (remember picture non supporté)  
`\cacheCommand{todo}[0{}m]{disable externalization}`

I am an overlay text

# Aperçu des fonctionnalités

- Gestion des images My node respecting baseline ❤️, avec overlay et depth.
- Détection semi-automatique des paquets/macros à charger.
- Gestion fine de la pureté (quand recompiler automatiquement).
- Automatiquement mettre en cache certaines commandes et environnements via:  
`\cacheEnvironment{forest}{\textrm{latex}, \add to preamble=\usepackage{forest}}`
- Possibilité de désactiver le cache pour certaines commandes/images (remember picture non supporté)  
`\cacheCommand{todo}[0]{\m}{\disable externalization}`
- Images en ligne via  
`\includegraphics[width=3cm]{https://placebear.com/400/300}.`

I am an overlay text

# Aperçu des fonctionnalités

- Gestion des images My node respecting baseline ❤️, avec overlay et depth.
- Détection semi-automatique des paquets/macros à charger.
- Gestion fine de la pureté (quand recompiler automatiquement).
- Automatiquement mettre en cache certaines commandes et environnements via:  
`\cacheEnvironment{forest}{\textrm{latex}, add to preamble=\usepackage{forest}}`
- Possibilité de désactiver le cache pour certaines commandes/images (remember picture non supporté)  
`\cacheCommand{todo}[0{}m]{\textrm{disable externalization}}`
- Images en ligne via  
`\includegraphicsWeb[width=3cm]{https://placebear.com/400/300}.`
- Special TikZit support.

I am an overlay text

# Aperçu des fonctionnalités

- Gestion des images My node respecting baseline ❤️, avec overlay et depth.
- Détection semi-automatique des paquets/macros à charger.
- Gestion fine de la pureté (quand recompiler automatiquement).
- Automatiquement mettre en cache certaines commandes et environnements via:  
`\cacheEnvironment{forest}{\textrm{latex}, add to preamble=\usepackage{forest}}`
- Possibilité de désactiver le cache pour certaines commandes/images (remember picture non supporté)  
`\cacheCommand{todo}[0]{m}{\disabled externalization}`
- Images en ligne via  
`\includegraphicsWeb[width=3cm]{https://placebear.com/400/300}.`
- Special TikZit support.
- Cache automatique (`\cacheTikz`) ou à la demande (`\begin{tikzpictureC}`).

I am an overlay text

# Aperçu des fonctionnalités

- Mettre en cache du code pour un cours é:

```
\begin{CacheMeCode}{python print code and result, set title={The for loop}}
    for name in ["Alice", "Bob"]:
        print(f"Hello {name}")
\end{CacheMeCode}
```

## The for loop

```
for name in ["Alice", "Bob"]:
    print(f"Hello {name}")
```

Output:

```
Hello Alice
Hello Bob
```

# Aperçu des fonctionnalités

- Mettre en cache du code pour un cours é:

```
\begin{CacheMeCode}{python print code and result, set title={The for loop}}
    for name in ["Alice", "Bob"]:
        print(f"Hello {name}")
\end{CacheMeCode}
```

## The for loop

```
for name in ["Alice", "Bob"]:
    print(f"Hello {name}")
```

Output:

```
Hello Alice
Hello Bob
```

- Générer des images (matplotlib, gnuplot, etc)

# Aperçu des fonctionnalités

- Configuration très modulaire avec placeholders

# Aperçu des fonctionnalités

- Configuration très modulaire avec placeholders
- Dépendances vers des fichiers externes

# Aperçu des fonctionnalités

- Configuration très modulaire avec placeholders
- Dépendances vers des fichiers externes
- Compiler encore plus rapidement en compilant des presets

# Aperçu des fonctionnalités

- Configuration très modulaire avec placeholders
- Dépendances vers des fichiers externes
- Compiler encore plus rapidement en compilant des presets
- Compilation en parallèle possible

# Aperçu des fonctionnalités

- Configuration très modulaire avec placeholders
- Dépendances vers des fichiers externes
- Compiler encore plus rapidement en compilant des presets
- Compilation en parallèle possible
- Suppression des éléments non-utilisés dans le cache

# Aperçu des fonctionnalités

- Configuration très modulaire avec placeholders
- Dépendances vers des fichiers externes
- Compiler encore plus rapidement en compilant des presets
- Compilation en parallèle possible
- Suppression des éléments non-utilisés dans le cache
- Gestion très générique des mises en cache

# Aperçu des fonctionnalités

- Configuration très modulaire avec placeholders
- Dépendances vers des fichiers externes
- Compiler encore plus rapidement en compilant des presets
- Compilation en parallèle possible
- Suppression des éléments non-utilisés dans le cache
- Gestion très générique des mises en cache
- ...

# Autres librairies

Voir aussi <https://github.com/sasozivanovic/memoize>, compromis différents:

- Avantages :
  - première compilation + rapide
  -
- Inconvénients :
  - spécifique à TikZ
  - **impure**

Ouvrons le capot !

demo.tex (/home/leo/Documents/Cours/Research/Presentations/2025\_11\_08\_-\_Gutenberg\_robust\_externalize/demo/demo.tex)

File Edit Options Buffers Tools Preview LaTeX Command YASnippet Jinx Ref Help

New File Open File Dired Kill Buffer Save Buffer Cut Copy Paste Undo | Latex View Bibtex Spell

```
\documentclass{article}
\usepackage{tikz}

\begin{document}

\begin{tikzpicture}
  \node[rounded corners, fill=green!20!white]{Hello};
\end{tikzpicture}

\end{document}
```

% Local Variables:

% TeX-command-extra-options: "-shell-escape -halt-on-error"

% End:

U:--- demo.tex All L4 (LaTeX/PS ε)

Wrote /home/leo/Documents/Cours/Research/Presentations/2025\_11\_08\_-\_Gutenberg\_robust\_externalize/demo/demo.tex

demo.tex (/home/leo/Documents/Cours/Research/Presentations/2025\_11\_08\_Gutenberg\_robust\_externalize/demo/demo.tex)

File Edit Options Buffers Tools Preview LaTeX Command YASnippet Jinx Ref Help

New File Open File Dired Kill Buffer Save Buffer Cut Copy Paste Undo Latex View Bibtex Spell

```
\documentclass{article}
\usepackage{tikz}
\usepackage{robust-externalize}
\begin{document}

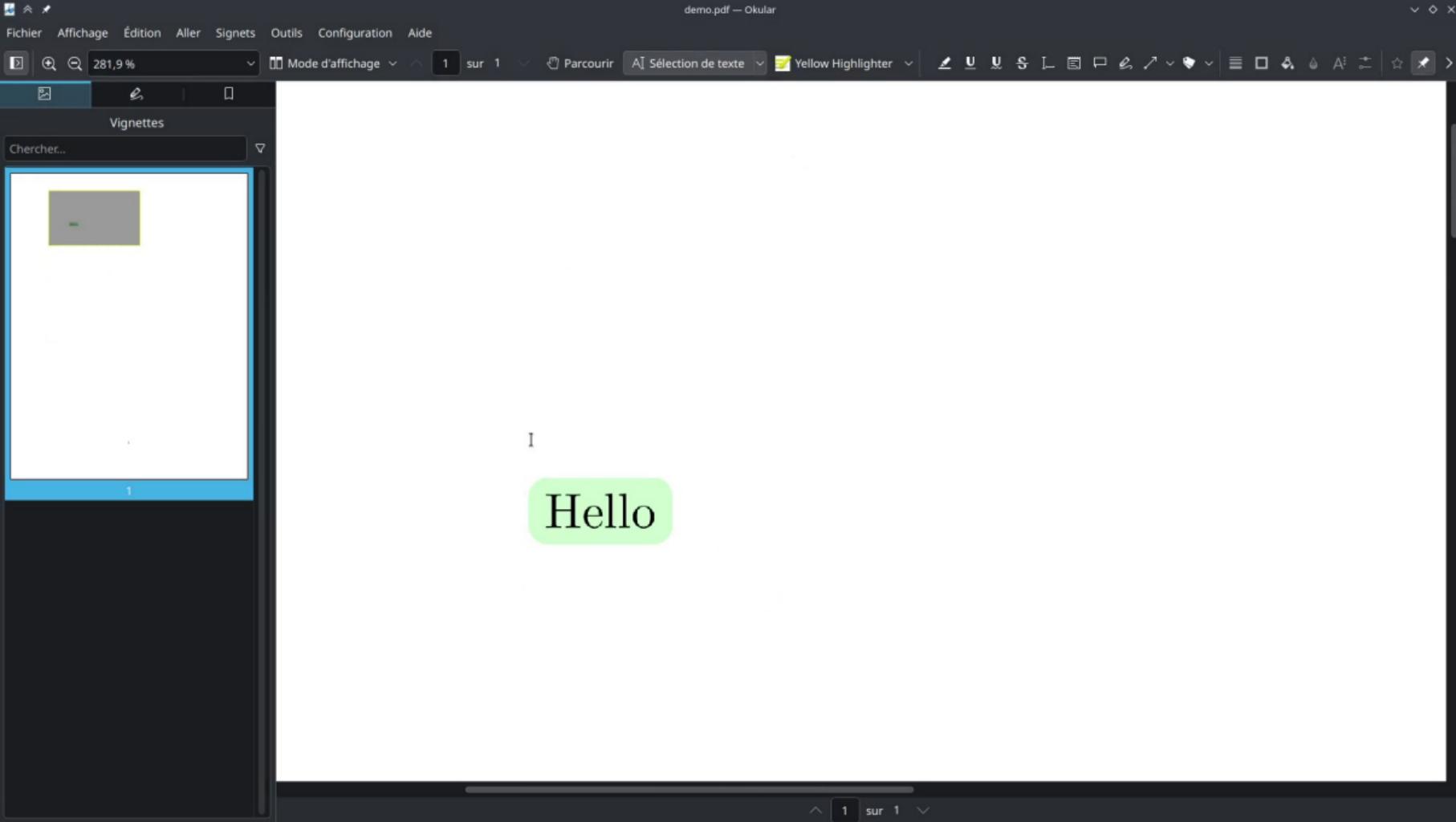
\begin{tikzpicture}
  \node[rounded corners, fill=green!20!white]{Hello};
\end{tikzpicture}

\end{document}
```

% Local Variables:

% TeX-command-extra-options: "-shell-escape -halt-on-error"

% End:



Nouvel onglet Scinder la vue robustExternalize : zsh — Konsole

Copier Coller Rechercher

```
robustExternalize : zsh — Konsole
```

rm robustExternalize/\*

zsh: sure you want to delete all 7 files in /home/leo/Documents/Cours/Research/Presentations/2025\_11\_08\_-\_Gutenberg\_robust\_externalize/demo/robustExternalize [yn]? y

cd robustExternalize

ls

robExt-65326EB87A65E882A8F5DE84810C3DC2.aux	robExt-65326EB87A65E882A8F5DE84810C3DC2-out.tex
robExt-65326EB87A65E882A8F5DE84810C3DC2-compilation.log	robExt-65326EB87A65E882A8F5DE84810C3DC2.pdf
robExt-65326EB87A65E882A8F5DE84810C3DC2.deps	robExt-65326EB87A65E882A8F5DE84810C3DC2.tex
robExt-65326EB87A65E882A8F5DE84810C3DC2.log	

robustExternalize : zsh — Konsole

Nouvel onglet Scinder la vue

robExt-65326EB87A65E882A8F5DE84810C3DC2.deps  
robExt-65326EB87A65E882A8F5DE84810C3DC2.log

robExt-65326EB87A65E882A8F5DE84810C3DC2.tex

```
* > cat robExt-65326EB87A65E882A8F5DE84810C3DC2.tex
\documentclass[,margin=30cm]{standalone}
\usepackage{tikz}
% most packages must be loaded before hyperref
% so we typically want to load hyperref here
% some packages must be loaded after hyperref

\begin{document}%
%% We save the height/depth of the content by using a savebox:
\newwrite\writeRobExt%
\immediate\openout\writeRobExt=\jobname-out.tex%
%
\newsavebox\boxRobExt%
\begin{lrbox}{\boxRobExt}%
\begin{tikzpicture}[] \node [rounded corners, fill=green!20!white]{Hello};\end{tikzpicture}%
\end{lrbox}%
\usebox{\boxRobExt}%
\immediate\write\writeRobExt{%
\string\def\string\robExtWidth{\the\wd\boxRobExt}%
\string\def\string\robExtHeight{\the\ht\boxRobExt}%
\string\def\string\robExtDepth{\the\dp\boxRobExt}%
}%
%
\end{document}
```

Nouvel onglet Scinder la vue

robExt-65326EB87A65E882A8F5DE84810C3DC2.deps  
robExt-65326EB87A65E882A8F5DE84810C3DC2.log

robExt-65326EB87A65E882A8F5DE84810C3DC2.tex

```
* > cat robExt-65326EB87A65E882A8F5DE84810C3DC2.tex
\documentclass[,margin=30cm]{standalone}
\nusepackage{tikz}
% most packages must be loaded before hyperref
% so we typically want to load hyperref here
% some packages must be loaded after hyperref

\begin{document}%
%% We save the height/depth of the content by using a savebox:
\newwrite\writeRobExt%
\immediate\openout\writeRobExt=\jobname-out.tex%
%
\newsavebox\boxRobExt%
\begin{lrbox}{\boxRobExt}%
\begin{tikzpicture}[] \node [rounded corners, fill=green!20!white]{Hello};\end{tikzpicture}%
\end{lrbox}%
\usebox{\boxRobExt}%
\immediate\write\writeRobExt{%
\string\def\string\robExtWidth{\the\wd\boxRobExt}%
\string\def\string\robExtHeight{\the\ht\boxRobExt}%
\string\def\string\robExtDepth{\the\dp\boxRobExt}%
}%
%
\end{document}
```

Nouvel onglet Scinder la vue

robExt-65326EB87A65E882A8F5DE84810C3DC2.deps  
robExt-65326EB87A65E882A8F5DE84810C3DC2.log

robExt-65326EB87A65E882A8F5DE84810C3DC2.tex

```
* > cat robExt-65326EB87A65E882A8F5DE84810C3DC2.tex
\documentclass[,margin=30cm]{standalone}
\usepackage{tikz}
% most packages must be loaded before hyperref
% so we typically want to load hyperref here
% some packages must be loaded after hyperref

\begin{document}%
%% We save the height/depth of the content by using a savebox:
\newwrite\writeRobExt%
\immediate\openout\writeRobExt=\jobname-out.tex%
%
\newsavebox\boxRobExt%
\begin{lrbox}{\boxRobExt}%
\begin{tikzpicture}[] \node [rounded corners, fill=green!20!white]{Hello};\end{tikzpicture}%
\end{lrbox}%
\usebox{\boxRobExt}%
\immediate\write\writeRobExt{%
\string\def\string\robExtWidth{\the\wd\boxRobExt}%
\string\def\string\robExtHeight{\the\ht\boxRobExt}%
\string\def\string\robExtDepth{\the\dp\boxRobExt}%
}%
%
\end{document}
```

Nouvel onglet Scinder la vue

robExt-65326EB87A65E882A8F5DE84810C3DC2.deps  
robExt-65326EB87A65E882A8F5DE84810C3DC2.log

robExt-65326EB87A65E882A8F5DE84810C3DC2.tex

```
* > cat robExt-65326EB87A65E882A8F5DE84810C3DC2.tex
\documentclass[,margin=30cm]{standalone}
\usepackage{tikz}
% most packages must be loaded before hyperref
% so we typically want to load hyperref here
% some packages must be loaded after hyperref

\begin{document}%
%% We save the height/depth of the content by using a savebox:
\newwrite\writeRobExt%
\immediate\openout\writeRobExt=\jobname-out.tex%
%
\newsavebox\boxRobExt%
\begin{lrbox}{\boxRobExt}%
\begin{tikzpicture}[] \node [rounded corners, fill=green!20!white]{Hello};\end{tikzpicture}%
\end{lrbox}%
\usebox{\boxRobExt}%
\immediate\write\writeRobExt{%
\string\def\string\robExtWidth{\the\wd\boxRobExt}%
\string\def\string\robExtHeight{\the\ht\boxRobExt}%
\string\def\string\robExtDepth{\the\dp\boxRobExt}%
}%
%
\end{document}
```

Copier Coller Rechercher

robExternalize : zsh — Konsole

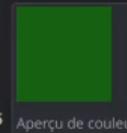
robExt-65326EB87A65E882A8F5DE84810C3DC2.tex

robExt-65326EB87A65E882A8F5DE84810C3DC2.log

robExternalize : zsh — Konsole

robExt-65326EB87A65E882A8F5DE84810C3DC2.tex

robExt-65326EB87A65E882A8F5DE84810C3DC2.log



Nouvel onglet Scinder la vue

robExt-65326EB87A65E882A8F5DE84810C3DC2.deps  
robExt-65326EB87A65E882A8F5DE84810C3DC2.log

robExt-65326EB87A65E882A8F5DE84810C3DC2.tex

```
* > cat robExt-65326EB87A65E882A8F5DE84810C3DC2.tex
\documentclass[,margin=30cm]{standalone}
\usepackage{tikz}
% most packages must be loaded before hyperref
% so we typically want to load hyperref here
% some packages must be loaded after hyperref

\begin{document}%
%% We save the height/depth of the content by using a savebox:
\newwrite\writeRobExt%
\immediate\openout\writeRobExt=\jobname-out.tex%
%
\newsavebox\boxRobExt%
\begin{lrbox}{\boxRobExt}%
\begin{tikzpicture}[] \node [rounded corners, fill=green!20!white]{Hello};\end{tikzpicture}%
\end{lrbox}%
\usebox{\boxRobExt}%
\immediate\write\writeRobExt{%
\string\def\string\robExtWidth{\the\wd\boxRobExt}%
\string\def\string\robExtHeight{\the\ht\boxRobExt}%
\string\def\string\robExtDepth{\the\dp\boxRobExt}%
}%
%
\end{document}
```

```
\usepackage {tikz}
% most packages must be loaded before hyperref
% so we typically want to load hyperref here

% some packages must be loaded after hyperref

\begin{document}%
%% We save the height/depth of the content by using a savebox:
\newwrite\writeRobExt%
\immediate\openout\writeRobExt=\jobname-out.tex%
%
\newsavebox\boxRobExt%
\begin{lrbox}{\boxRobExt}%
 \begin{tikzpicture}[] \node [rounded corners, fill=green!20!white]{Hello};\end{tikzpicture}%
\end{lrbox}%
\usebox{\boxRobExt}%
\immediate\write\writeRobExt{%
 \string\def\string\robExtWidth{\the\wd\boxRobExt}%
 \string\def\string\robExtHeight{\the\ht\boxRobExt}%
 \string\def\string\robExtDepth{\the\dp\boxRobExt}%
}%
%
\end{document}
```

```
~ > cat robExt-65326EB87A65E882A8F5DE84810C3DC2.deps .....
```

command,pdfflatex -halt-on-error "\_\_ROBEXT\_SOURCE\_FILE\_\_"  
8DBD2711149029B8AA70E0B6A6D1B181,

```
~ > cat robustExternalize .....
```

```
\usepackage {tikz}
% most packages must be loaded before hyperref
% so we typically want to load hyperref here

% some packages must be loaded after hyperref

\begin{document}%
%% We save the height/depth of the content by using a savebox:
\newwrite\writeRobExt%
\immediate\openout\writeRobExt=\jobname-out.tex%
%
\newsavebox\boxRobExt%
\begin{lrbox}{\boxRobExt}%
 \begin{tikzpicture}[] \node [rounded corners, fill=green!20!white]{Hello};\end{tikzpicture}%
\end{lrbox}%
\usebox{\boxRobExt}%
\immediate\write\writeRobExt{%
 \string\def\string\robExtWidth{\the\wd\boxRobExt}%
 \string\def\string\robExtHeight{\the\ht\boxRobExt}%
 \string\def\string\robExtDepth{\the\dp\boxRobExt}%
}%
%
\end{document}
```

```
 cat robExt-65326EB87A65E882A8F5DE84810C3DC2.deps
command,pdfflatex -halt-on-error "__ROBEXT_SOURCE_FILE__"
8BDB2711149029B8AA70E0B6A6D1B181,
```

```
robustExternalize : zsh — Konsole
robustExternalize : zsh — Konsole
robustExternalize : zsh — Konsole
```

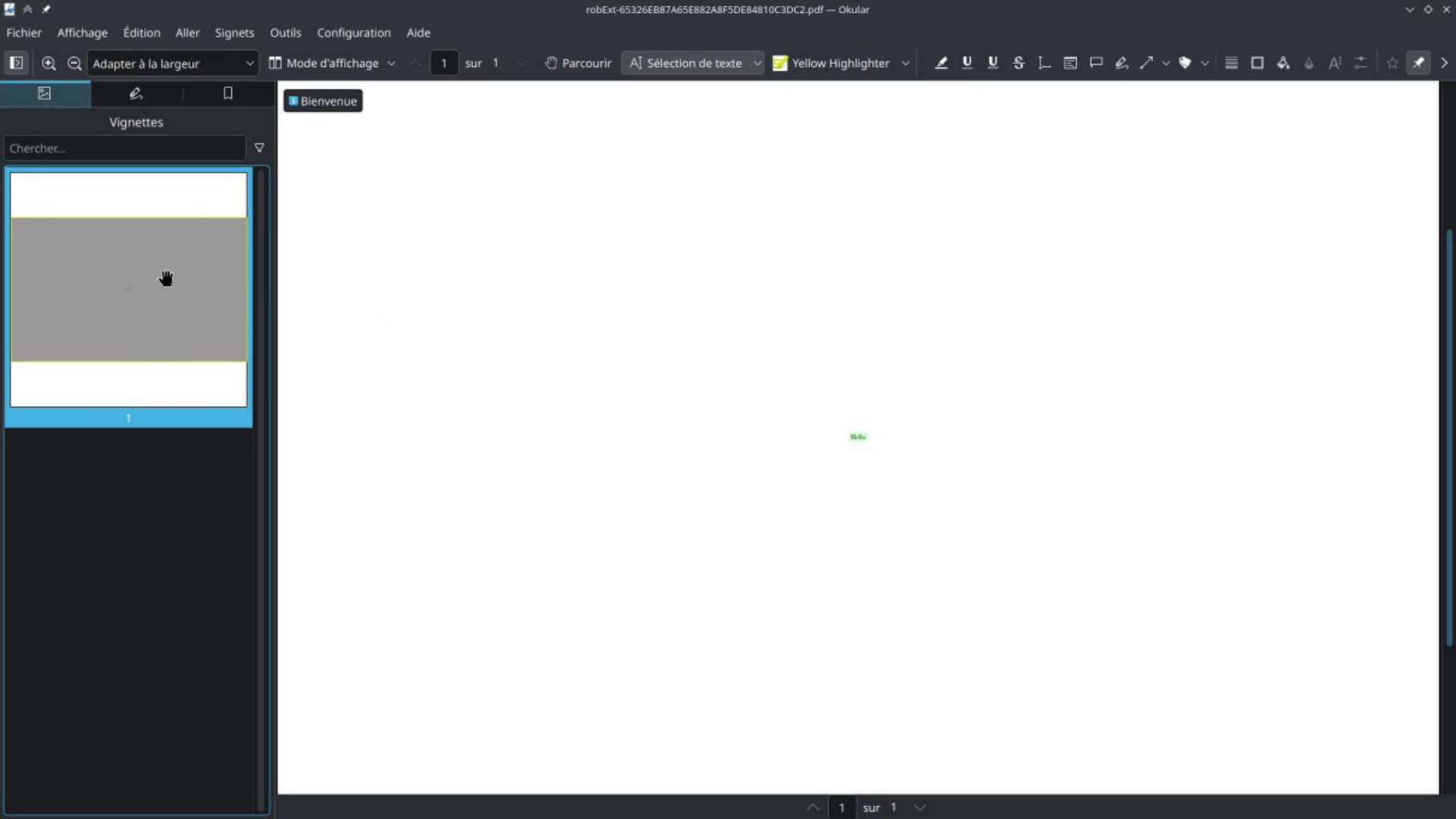
```
\usepackage {tikz}
% most packages must be loaded before hyperref
% so we typically want to load hyperref here

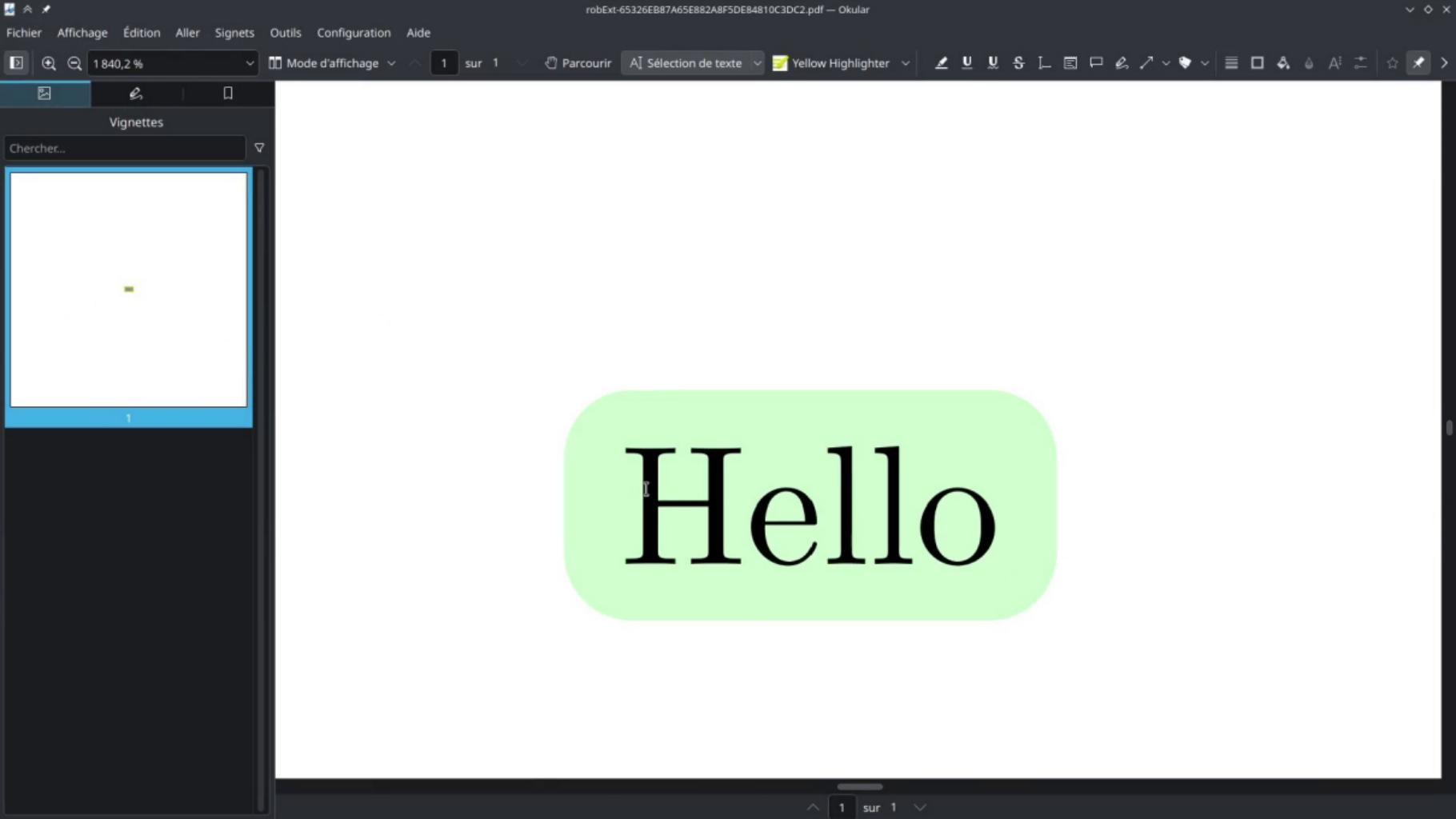
% some packages must be loaded after hyperref

\begin{document}%
%% We save the height/depth of the content by using a savebox:
\newwrite\writeRobExt%
\immediate\openout\writeRobExt=\jobname-out.tex%
%
\newsavebox\boxRobExt%
\begin{lrbox}{\boxRobExt}%
 \begin{tikzpicture}[] \node [rounded corners, fill=green!20!white]{Hello};\end{tikzpicture}%
\end{lrbox}%
\usebox{\boxRobExt}%
\immediate\write\writeRobExt{%
 \string\def\string\robExtWidth{\the\wd\boxRobExt}%
 \string\def\string\robExtHeight{\the\ht\boxRobExt}%
 \string\def\string\robExtDepth{\the\dp\boxRobExt}%
}%
%
\end{document}
```

```
 cat robExt-65326EB87A65E882A8F5DE84810C3DC2.deps
command,pdfflatex -halt-on-error "__ROBEXT_SOURCE_FILE__"
8BDB2711149029B8AA70E0B6A6D1B181,
```

```
 cat robExt-65326EB87A65E882A8F5DE84810C3DC2.deps
command,pdfflatex -halt-on-error "__ROBEXT_SOURCE_FILE__"
8BDB2711149029B8AA70E0B6A6D1B181,
```





```
command,pdflatex -halt-on-error "__ROBEXT_SOURCE_FILE__"  
8BDB2711149029B8AA70E0B6A6D1B181,
```

```
ls  
robExt-65326EB87A65E882A8F5DE84810C3DC2.aux robExt-65326EB87A65E882A8F5DE84810C3DC2-out.tex  
robExt-65326EB87A65E882A8F5DE84810C3DC2-compilation.log robExt-65326EB87A65E882A8F5DE84810C3DC2.pdf  
robExt-65326EB87A65E882A8F5DE84810C3DC2.deps robExt-65326EB87A65E882A8F5DE84810C3DC2.tex  
robExt-65326EB87A65E882A8F5DE84810C3DC2.log
```

```
xdg-open robExt-65326EB87A65E882A8F5DE84810C3DC2.pdf
```

```
cat robExt-65326EB87A65E882A8F5DE84810C3DC2.aux  
\relax  
\gdef \@abspage@last{1}
```

```
cat robExt-65326EB87A65E882A8F5DE84810C3DC2-out.tex  
\def\robExtWidth{29.16599pt}\def\robExtHeight{13.61038pt}\def\robExtDepth{0.0pt}
```

```
█
```

```
command,pdflatex -halt-on-error "__ROBEXT_SOURCE_FILE__"  
8BDB2711149029B8AA70E0B6A6D1B181,
```

```
ls
```

```
robExt-65326EB87A65E882A8F5DE84810C3DC2.aux          robExt-65326EB87A65E882A8F5DE84810C3DC2-out.tex  
robExt-65326EB87A65E882A8F5DE84810C3DC2-compilation.log  robExt-65326EB87A65E882A8F5DE84810C3DC2.pdf  
robExt-65326EB87A65E882A8F5DE84810C3DC2.deps        robExt-65326EB87A65E882A8F5DE84810C3DC2.tex  
robExt-65326EB87A65E882A8F5DE84810C3DC2.log
```

```
xdg-open robExt-65326EB87A65E882A8F5DE84810C3DC2.pdf
```

```
cat robExt-65326EB87A65E882A8F5DE84810C3DC2.aux
```

```
\relax  
\gdef \@abspage@last{1}
```

```
cat robExt-65326EB87A65E882A8F5DE84810C3DC2-out.tex
```

```
\def\robExtWidth{29.16599pt}\def\robExtHeight{13.61038pt}\def\robExtDepth{0.0pt}
```

```
█
```

Nouvel onglet Scinder la vue robustExternalize : zsh — Konsole

Copier Coller Rechercher

```
% most packages must be loaded before hyperref
% so we typically want to load hyperref here

% some packages must be loaded after hyperref

\begin{document}%
%% We save the height/depth of the content by using a savebox:
\newwrite\writeRobExt%
\immediate\openout\writeRobExt=\jobname-out.tex%
%
\newsavebox\boxRobExt%
\begin{lrbox}{\boxRobExt}%
 \begin{tikzpicture}[] \node [rounded corners, fill=green!20!white]{Hello};\end{tikzpicture}%
\end{lrbox}%
\usebox{\boxRobExt}%
\immediate\write\writeRobExt{%
 \string\def\string\robExtWidth{\the\wd\boxRobExt}%
 \string\def\string\robExtHeight{\the\ht\boxRobExt}%
 \string\def\string\robExtDepth{\the\dp\boxRobExt}%
}%
%
\end{document}
```

```
* > D~/Cours/R/P/2025_11_08_-_Gutenberg_robust_externalize/demo/robustExternalize
> cat robExt-65326EB87A65E882A8F5DE84810C3DC2.deps
command,pdflatex -halt-on-error "__ROBEXT_SOURCE_FILE__"
8BDB2711149029B8AA70E0B6A6D1B181,
```

```
* > D~/Cours/R/P/2025_11_08_-_Gutenberg_robust_externalize/demo/robustExternalize
> ls
robExt-65326EB87A65E882A8F5DE84810C3DC2.aux
robExt-65326EB87A65E882A8F5DE84810C3DC2-out.tex
```

demo.tex (/home/leo/Documents/Cours/Research/Presentations/2025\_11\_08\_Gutenberg\_robust\_externalize/demo/demo.tex)

File Edit Options Buffers Tools Preview LaTeX Command YASnippet Jinx Ref Help

New File Open File Dired Kill Buffer Save Buffer Cut Copy Paste Undo Latex View Bibtex Spell

```
\documentclass{article}
\usepackage{tikz}
\usepackage{robust-externalize}
\begin{document}

\begin{tikzpicture}
  \node[rounded corners, fill=green!20!white]{Hello};
\end{tikzpicture}

\end{document}
```

% Local Variables:  
% TeX-command-extra-options: "-shell-escape -halt-on-error"  
% End:

U:++- demo.tex All L6 (LaTeX/PS ε)

demo.tex (/home/leo/Documents/Cours/Research/Presentations/2025\_11\_08\_Gutenberg\_robust\_externalize/demo/demo.tex)

File Edit Options Buffers Tools Preview LaTeX Command YASnippet Jinx Ref Help

New File Open File Dired Kill Buffer Save Buffer Cut Copy Paste Undo Latex View Bibtex Spell

```
\documentclass{article}
\usepackage{tikz}
\usepackage{robust-externalize}
\cacheTikz
\begin{document}

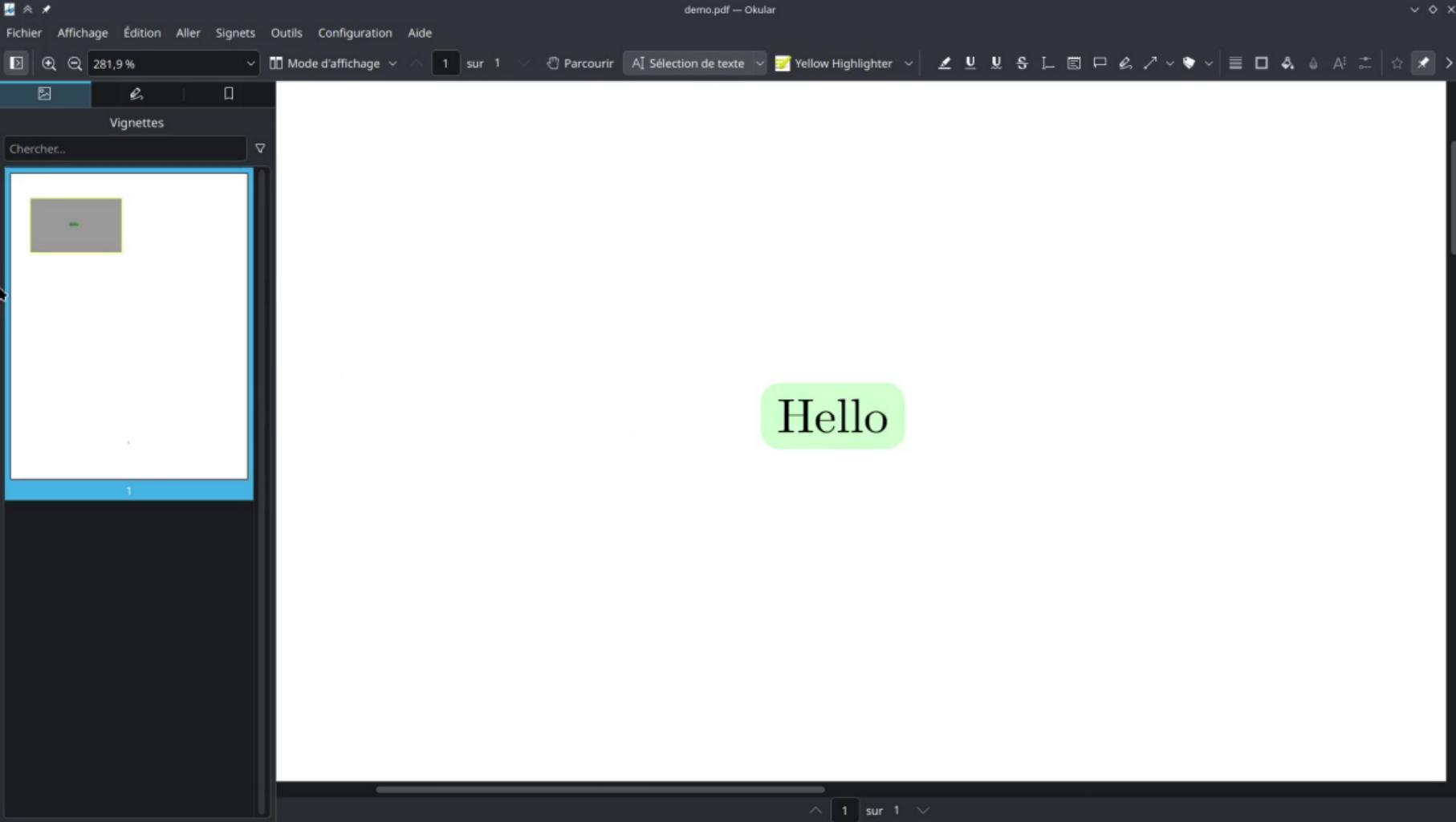
\begin{tikzpicture}
  \node[rounded corners, fill=green!20!white]{Hello};
\end{tikzpicture}

\end{document}
```

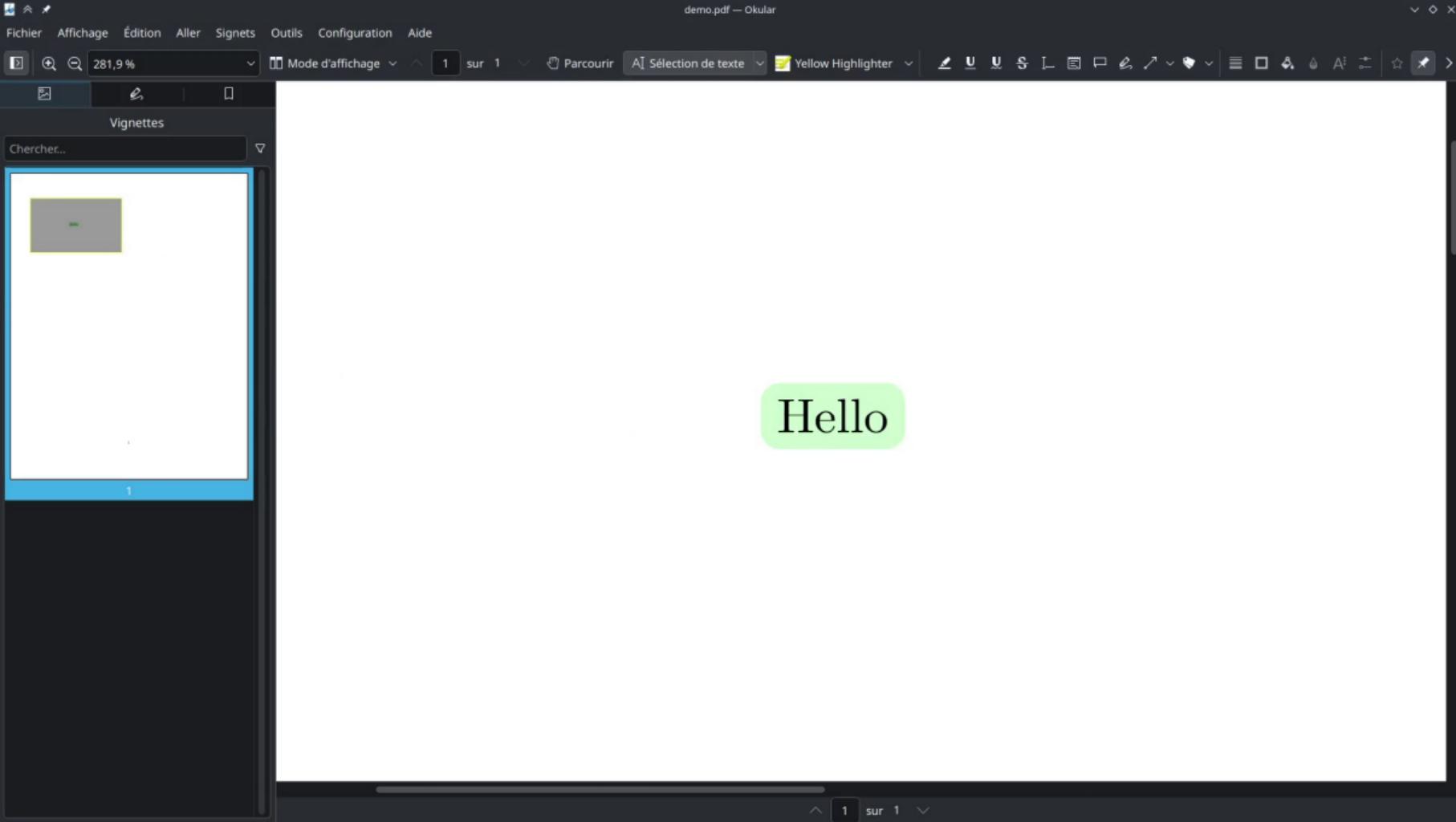
% Local Variables:

% TeX-command-extra-options: "-shell-escape -halt-on-error"

% End:



Hello



# Presets

**Preset** = ensemble de commandes (pgfkeys) qui visent à **configurer** la mise en cache (template, compilation, dépendances, et inclusion)

```
\robExtConfigure{  
    new preset={my online img}{  
        set template={}, % no code needed  
        set compilation command={  
            wget "__ESCAPEROBEXT_MAIN_CONTENT__" -O "__ESCAPEROBEXT_OUTPUT_PDF__"},  
        custom include command=%  
            \evalPlaceholder%  
            \includegraphics[__MY_OPTIONS__]{%  
                \robExtAddCachePath{__ESCAPEROBEXT_OUTPUT_PDF__}%  
            }%  
        }%  
    }%  
}
```

# Presets

On peut également **modifier** un preset:

```
\robExtConfigure{  
    add to preset={tikz}{  
        add to preamble={\usepackage{tikz}}[shadows],  
    },  
}
```

# Presets

Les presets peuvent étendre d'autres presets, e.g. dans la librairie tikz se base sur latex, et tikzpicture sur tikz:

```
\robExtConfigure{
    tikz/.style={
        latex,
        add to preamble={\usepackage{tikz}},
    },
    tikzpicture/.style={
        tikz,
        set placeholder no import={__ROBEXT_MAIN_CONTENT__}%
            {\begin{tikzpicture}__ROBEXT_MAIN_CONTENT_ORIG_\end{tikzpicture}},
    },
}
```

# Placeholders

La **template** (fichier de base à compiler), la commande de compilation, et possiblement la commande d'inclusion sont créés à partir de **placeholders** ( $\approx$  variable/macro) récursivement remplacés au moment de la compilation:

```
\setPlaceholder{__ROBEXT_LATEX_COMPILATION_COMMAND__}%
  {__ROBEXT_LATEX_ENGINE__ __ROBEXT_LATEX_COMPILATION_COMMAND_OPTIONS__%
   " __ROBEXT_SOURCE_FILE__"}
\setPlaceholder{__ROBEXT_LATEX_COMPILATION_COMMAND_OPTIONS__}%
  {-halt-on-error -interaction=nonstopmode}
\setPlaceholder{__ROBEXT_LATEX_ENGINE__}{pdflatex}
```

# Placeholders

Exemple du placeholder de template L<sup>A</sup>T<sub>E</sub>X utilisé dans la librairie :

```
\begin{RobExtPlaceholderFromCode}{__ROBEXT_LATEX__}
  \documentclass[__ROBEXT_LATEX_OPTIONS__]{__ROBEXT_LATEX_DOCUMENT_CLASS__}
  __ROBEXT_LATEX_PREAMBLE__
  % most packages must be loaded before hyperref
  % so we typically want to load hyperref here
  __ROBEXT_LATEX_PREAMBLE_HYPERREF__
  % some packages must be loaded after hyperref
  __ROBEXT_LATEX_PREAMBLE_AFTER_HYPERREF__
  \begin{document}%
  __ROBEXT_LATEX_MAIN_CONTENT_WRAPPED__
  \end{document}
\end{RobExtPlaceholderFromCode}
```

# Placeholders

Ces placeholders sont alors appelés et configurés dans les presets, par exemple pour  $\text{\LaTeX}$  nous faisons :

```
\robExtConfigure{
    latex/.style={
        set template=__ROBEXT_LATEX__,
        set compilation command=__ROBEXT_LATEX_COMPILATION_COMMAND__,
        use latexmk/.style={
            set placeholder=__ROBEXT_LATEX_ENGINE__}{latexmk},
        },
        % ...
    }
}
```

# Placeholders particuliers

Quelques placeholders avec un **rôle particulier** :

- `__ROBEXT_MAIN_CONTENT__` : contient le code tapé par l'utilisateur (e.g. image TikZ...)
- `__ROBEXT_SOURCE_FILE__` : chemin du fichier créé par la template (forme `robExt-somehash.tex`)
- `__ROBEXT_OUTPUT_PDF__` : chemin du fichier à créer si la compilation a réussi (forme `robExt-somehash.pdf`)
- `__ROBEXT_OUTPUT_PREFIX__` : préfix à utiliser pour créer d'autres fichiers (forme `robExt-somehash`)
- `__ROBEXT_WAY_BACK__` : par défaut vaut `.. /`, c'est le chemin pour aller du dossier de cache (`robustExternalize` par défaut) au sources de l'utilisateur

# Utiliser un preset

Pour **utiliser un preset** :

```
\begin{CacheMeCode}{mon preset, commandes additionnelles}
    code utilisateur
\end{CacheMeCode}
```

Si le code utilisateur contient du  $\text{\LaTeX}$ , on peut utiliser CacheMe à la place ou  
`\cacheMe[preset, autres commandes]{code}` :

- CacheMe parse le **contenu en mode  $\text{\LaTeX}$**  (et donc supprime les commentaires, les lignes vides, ajoute des espaces après les macros etc) mais peut-être utilisé partout
- CacheMeCode garde le **contenu intacte**, mais ne peut pas vraiment être utilisé dans des macros et certains environnements (**limitations  $\text{\LaTeX}$** )

# Utiliser un preset

Autre manière pour utiliser un preset:

- `\cacheCommand{mymacro}[xparse arg spec]{my preset, other options}` va automatiquement appliquer le preset/commandes sur `\mymacro`
- `\cacheEnvironment{myenv}{my preset, other options}` va automatiquement appliquer le preset/commandes sur `\begin{myenv}`

⇒ comme ça que l'on cache TikZ :

```
\robExtCacheEnvironment{tikzpicture}{tikzpicture}
```

# Utiliser un preset

Autre manière pour utiliser un preset:

- `\cacheCommand{mymacro}[xparse arg spec]{my preset, other options}` va automatiquement appliquer le preset/commandes sur `\mymacro`
- `\cacheEnvironment{myenv}{my preset, other options}` va automatiquement appliquer le preset/commandes sur `\begin{myenv}`

⇒ comme ça que l'on cache TikZ :

```
\robExtCacheEnvironment{tikzpicture}{tikzpicture}
```

Ceci modifie également les commandes/macros en rajoutant une **option pour rajouter des options** :

```
\begin{tikzpicture}<disable externalization>[remember picture,overlay]
  \node at (current page.center){Foo};
\end{tikzpicture}
```

# Dépendance

**Motivation** : recompiler images quand un fichier change

```
\begin{CacheMe}{latex,
    add dependencies={common_inputs.tex},
    add to preamble={\input{__ROBEXT_WAY_BACK_common_inputs.tex}}}
    The answer is \myValueDefinedInCommonInputs.
\end{CacheMe}
```

# Utilisations

# Utilisations

→ TikZ

Considérons :

```
\cacheTikz
\def\someName{Alice}
\begin{tikzpicture}
  \node[rounded corners, fill=green!20!white]{Hey \someName};
\end{tikzpicture}
```

**Problème** : ça ne compilera **PAS !!**

Considérons :

```
\cacheTikz
\def\someName{Alice}
\begin{tikzpicture}
  \node[rounded corners, fill=green!20!white]{Hey \someName};
\end{tikzpicture}
```

**Problème** : ça ne compilera **PAS !!**

**Raison** : **\someName n'est pas recopié** dans le fichier externe qui est compilé

# Solution : forwarder les macros

## Solutions :

- Manuelle :

```
\cacheTikz
\def\someName{Alice}
\begin{tikzpicture}<forward=\someName>
    \node[rounded corners, fill=green!20!white]{Hey \someName};
\end{tikzpicture}
```

# Solution : forwarder les macros

## Solutions :

- Manuelle :

```
\cacheTikz
\def\someName{Alice}
\begin{tikzpicture}<forward=\someName>
    \node[rounded corners, fill=green!20!white]{Hey \someName};
\end{tikzpicture}
```

- Automatique (spécifier quels macros forwarder, détection automatique) :

```
\cacheTikz                                     (not needed after 4.0)
\robExtConfigure{add to preset={tikz}{auto forward}}
\defAutoForward\someName{Alice}
\begin{tikzpicture}
    \node[rounded corners, fill=green!20!white]{Hey \someName};
\end{tikzpicture}
```

# Solution : forwarder les macros

## Solutions (not recommended but possibly simpler) :

- Trop pure (recompile toutes les images quand changement) + code dupliqué :

```
\robExtConfigure{
    add to preset={tikz}{
        add to preamble={
            \def\someName{Alice}
        },
    },
}
\def\someName{Alice}
\begin{tikzpicture}
    \node[rounded corners, fill=green!20!white]{Hey \someName};
\end{tikzpicture}
```

# Solution : forwarder les macros

## Solutions (not recommended but possibly simpler) :

- Trop pure (recompile tout quand changement) mais **très simple** (**nouveau** version 4.0) :

```
\begin{CacheMeCode}{run here and in preamble of cached files}
  \def\someName{Alice}
\end{CacheMeCode}

\begin{tikzpicture}
  \node[rounded corners, fill=green!20!white]{Hey \someName};
\end{tikzpicture}
```

# Solution : forwarder les macros

## Solutions (not recommended but possibly simpler) :

- Impure mais simple (**nouveau** version 4.0) :

```
\robExtConfigure{
    add to preset={latex}{
        add to preamble={
            \input{impure.tex}
        },
    }
}

\begin{CacheMeCode}{run here and impurely create or add to file=impure.tex}
    \def\someName{Alice}
\end{CacheMeCode}

\begin{tikzpicture}
    \node[rounded corners, fill=green!20!white]{Hey \someName};
\end{tikzpicture}
```

On force la recompilation avec `\begin{tikzpicture}<recompile>`.

# Solution : forwarder les macros

Nouveau version 4.0: possible de faire comme `\defAutoForward` mais pour transmettre un groupe de commandes :

```
\begin{CacheMeCode}{run here and if macro present={\name,\firstName}{myenvA}}
  \newcommand{\firstName}{Alice}
  \newcommand{\lastName}{Bar}
  \newcommand{\name}{\firstName \lastName}
  \NewDocumentEnvironment{myenvA}{}{\firstName}{\lastName}
\end{CacheMeCode}

\begin{tikzpicture}
  \node[rounded corners, fill=green!20!white]{Hey \name};
\end{tikzpicture}
```

# Et pour le code non- $\text{\LaTeX}$ ?

Également possible d'**exécuter une configuration arbitraire** quand un **macro/mot/regex/...** est présent

```
\robExtConfigure{
    add to preset={my python} {
        python print code and result,
        if matches={cos()} {add import={from math import cos}},
        if matches={sin()} {add import={from math import sin}},
    },
}
\begin{CacheMeCode}{my python}
    print(cos(1)+sin(2))
\end{CacheMeCode}
```

# What about non-L<sup>A</sup>T<sub>E</sub>X code?

**Result :**

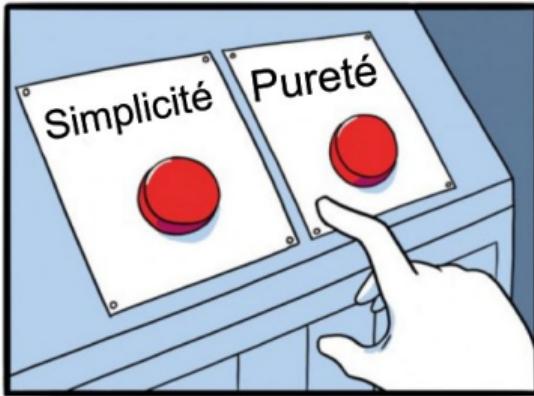
Python code

```
print(cos(1)+sin(2))
```

Output:

1.4495997326938215

# Simplicité vs pureté



# Utilisations

→ Python

# Module python : pratique pour faire des cours

```
\begin{CacheMeCode}{python print code and result, set title={The for loop}}
    for name in ["Alice", "Bob"]:
        print(f"Hello {name}")
\end{CacheMeCode}
```

## The for loop

```
for name in ["Alice", "Bob"]:
    print(f"Hello {name}")
```

Output:

Hello Alice  
Hello Bob

Aller plus loin

Voici quelques applications **non-publiées**<sup>1</sup> dans cette librairie, mais qui se basent sur elle :-)

---

<sup>1</sup>Pour le moment...

# Aller plus loin

→ Annoter ses slides avec xournal++

# Annoter ses slides avec xournal++

**Motivations** : faire des dessins TikZ, c'est long

⇒ But = rapidement **annoter ses slides** avec une tablette graphique via xournal++

# Annoter ses slides avec xournal++

**Motivations** : faire des dessins TikZ, c'est long

⇒ But = rapidement **annoter ses slides** avec une tablette graphique via xournal++

mais rigolo



# Annoter ses slides avec xournal++

**Motivations** : faire des dessins TikZ, c'est long

→ But = rapidement **annoter ses slides** avec une tablette graphique via xournal++

→ et avec "duplices" on peut simuler des overlays !

mais rigolo

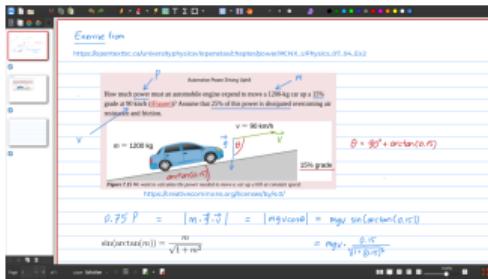


# Annoter ses slides avec xournal++

**Motivations** : faire des dessins TikZ, c'est long

→ But = rapidement **annoter ses slides** avec une tablette graphique via xournal++

→ et avec "dupliques" on peut simuler des overlays !



et même ajouter des images !

```
%% FRAME_START
\subsection{Annoter ses slides avec xournal++}
%% FRAME_STOP
```

□

```
\begin{frame}{Annoter ses slides avec xournal++}
  \importantB{Motivations} : faire des dessins Ii\emph{k}Z, c'est long

  \bigskip

  $ \Rightarrow $ But = rapidement \importantB{annoter ses slides} avec une tablette graphique via xournal++
\end{frame}
```

```
%% FRAME_START
\subsection{Annoter ses slides avec xournal++}
%% FRAME_STOP

%% FRAME_START
\begin{CacheMeCode}{xournal=demo_slide,edit}

\begin{frame}{Annoter ses slides avec xournal++}
\importantB{Motivations} : faire des dessins  $\text{Ti}\text{\texttt{emph}}{k}Z$ , c'est long

\bigs skip

\$ \Rightarrow \$ But = rapidement \importantB{annoter ses slides} avec une tablette graphique via xournal++
\end{frame}

\end{CacheMeCode}
%% FRAME_STOP
```

```
%% FRAME_START
\subsection{Annoter ses slides avec xournal++}
%% FRAME_STOP

%% FRAME_START
\begin{CacheMeCode}{xournal=demo_slide,edit}

\begin{frame}{Annoter ses slides avec xournal++}
\importantB{Motivations} : faire des dessins  $\text{Ti}\text{\emph{k}}Z$ , c'est long

\bigs skip

\$ \Rightarrow \$ But = rapidement \importantB{annoter ses slides} avec une tablette graphique via xournal++
\end{frame}

\end{CacheMeCode}
%% FRAME_STOP
```

Fichier Editer Affichage Navigation Journal Outils Greffon Aide

demo\_slide.pdf - Xournal++

Noto Sans Regular 30

Annoter ses slides avec xournal++

**Motivations** : faire des dessins TikZ, c'est long

⇒ But = rapidement **annoter ses slides** avec une tablette graphique via xournal++

Léo Colisson | 1

Page 1 sur 1, Page PDF 1 Calque Calque 1 269%

Fichier Editer Affichage Navigation Journal Outils Greffon Aide

\*demo\_slide.pdf - Xournal++

Noto Sans Regular 30

Annoter ses slides avec xournal++

**Motivations** : faire des dessins TikZ, c'est long

⇒ But = rapidement **annoter ses slides** avec une tablette graphique via xournal++

mais rigolo

Léo Colisson | 1

Page 1 sur 1, Page PDF 1 Calque Calque 1 269%

Fichier Editer Affichage Navigation Journal Outils Greffon Aide

\*demo\_slide.pdf - Xournal++

Noto Sans Regular 30

1 2

# Annoter ses slides avec xournal++

je via

**Motivations :** faire des dessins TikZ, c'est long

⇒ But = rapidement **annoter ses slides** avec une tablette graphique via xournal++

et avec "duplices" on peut simuler des overlays !

mais rigolo

Léo Colisson | 1

Page 2 sur 2, Page PDF 1 Calque Calque 1 269%

Fichier Editer Affichage Navigation Journal Outils Greffon Aide

\*demo\_slide.pdf - Xournal++

Noto Sans Regular 30

Xournal++ - Xournal++ — Mozilla Firefox

https://xournalpp.github.io

Blenderpoint web rea... Slack CWI | modern-c... 2022\_efficient\_OT Personal Kanboard Codeberg-CI/request-... Reanimate Other Bookmarks

xournalpp/xournalpp v1.28 15.7k 956

Exercise from https://opentextbc.ca/universityphysics/openstax/chapter/power/#CNX\_UPhysics\_07\_04\_Ex

Automotive Power Driving Uphill

How much power must an automobile engine expend to move a 1200-kg car up a 15% grade at 90 km/h (Figure)? Assume that 25% of this power is dissipated overcoming air resistance and friction.

$m = 1200 \text{ kg}$

$v = 90 \text{ km/h}$

$\theta = 90^\circ + \arctan(0.15)$

$P = m g v \sin(\theta)$

Figure 7.15 We want to calculate the power needed to move a car up a hill at constant speed.

https://creativecommons.org/licenses/by/4.0/

$0.75 P = |m \cdot \vec{g} \cdot \vec{v}| = |mg v \cos\theta| = mgv \sin(\arctan(0.15))$

$\sin(\arctan(m)) = \frac{m}{\sqrt{1+m^2}}$

$= mgv \cdot \frac{0.15}{\sqrt{1+(0.15)^2}}$

1 2

ue via

140%

Page 2 sur 2, Page PDF 1 Calque Calque 1

269%

Fichier Editer Affichage Navigation Journal Outils Greffon Aide

\*demo\_slide.pdf - Xournal++

Noto Sans Regular 30

1 2

# Annoter ses slides avec xournal++

je via

**Motivations :** faire des dessins TikZ, c'est long

⇒ But = rapidement **annoter ses slides** avec une tablette graphique via xournal++

et avec "duplices" on peut simuler des overlays !

mais rigolo

Léo Colisson | 1

Page 2 sur 2, Page PDF 1 Calque Calque 1 269%

Fichier Editer Affichage Navigation Journal Outils Greffon Aide

\*demo\_slide.pdf - Xournal++

Noto Sans Regular 30

1 2

# Annoter ses slides avec xournal++

je via

mais rigolo ☺

**Motivations :** faire des dessins TikZ, c'est long

⇒ But = rapidement **annoter ses slides** avec une tablette graphique via xournal++

et avec "duplices" on peut simuler des overlays !

et même g

Léo Colisson | 1

Page 2 sur 2, Page PDF 1 Calque Calque 1 269%

The image shows a screenshot of the Xournal++ application interface. The main window displays a presentation slide titled "Annoter ses slides avec xournal++". On the slide, there is handwritten text: "mais rigolo" with a smiley face, "je via", and another "mais rigolo" with a smiley face. Below the slide, there is a text block with the following content:

**Motivations :** faire des dessins TikZ, c'est long

⇒ But = rapidement **annoter ses slides** avec une tablette graphique via xournal++

et avec "duplices" on peut simuler des overlays !

et même g

At the bottom of the slide, there is a small screenshot of a presentation slide showing some mathematical calculations. The Xournal++ interface includes a toolbar at the top with various drawing tools, and a status bar at the bottom showing page information and zoom levels.

Fichier Editer Affichage Navigation Journal Outils Greffon Aide

\*demo\_slide.pdf - Xournal++

Noto Sans Regular 30

1 2

# Annoter ses slides avec xournal++

je via

mais rigolo ☺

**Motivations :** faire des dessins TikZ, c'est long

⇒ But = rapidement **annoter ses slides** avec une tablette graphique via xournal++

et avec "duplices" on peut simuler des overlays !

et même ajou

Léo Colisson | 1

Page 2 sur 2, Page PDF 1 Calque Calque 1 269%

Fichier Editer Affichage Navigation Journal Outils Greffon Aide

\*demo\_slide.pdf - Xournal++

Noto Sans Regular 30

1 2 3

# Annoter ses slides avec xournal++

**Motivations :** faire des dessins TikZ, c'est long

→ But = rapidement **annoter ses slides** avec une tablette graphique via xournal++

→ et avec "duplices" on peut simuler des overlays !

→ et même ajouter des images !

Léo Colisson | 1

269%

Fichier Editer Affichage Navigation Journal Outils Greffon Aide

\*demo\_slide.pdf - Xournal++

Noto Sans Regular 30

1 2 3

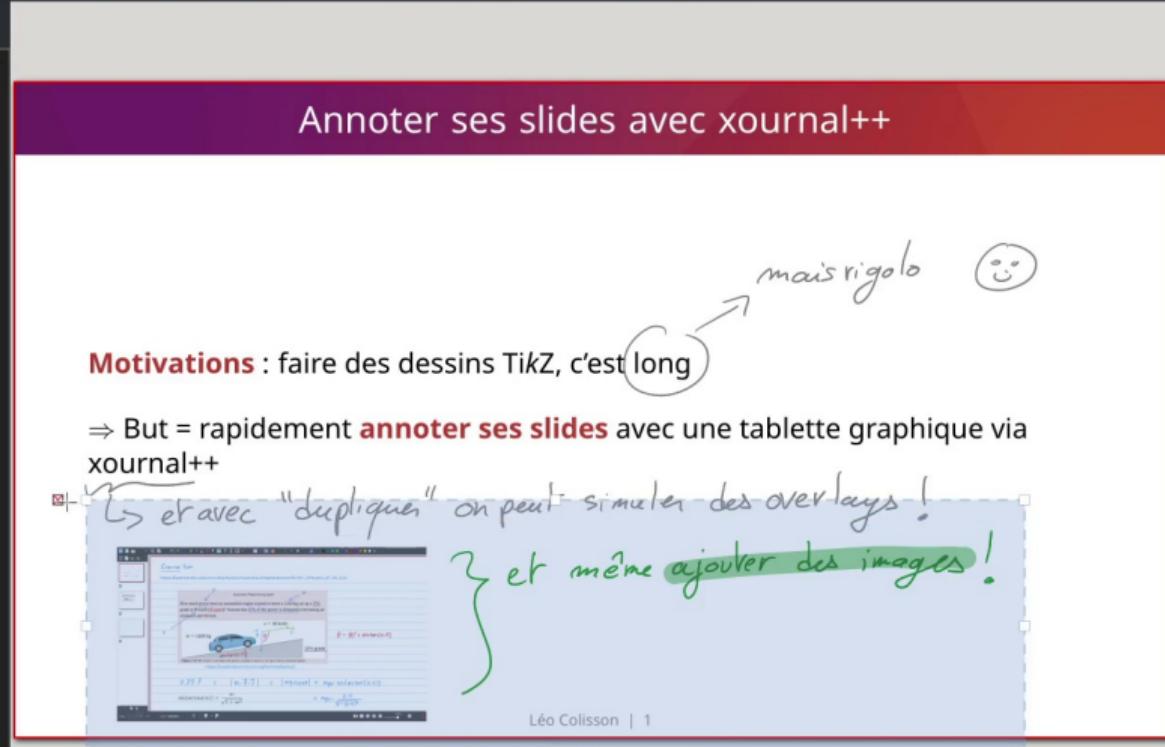
# Annoter ses slides avec xournal++

**Motivations :** faire des dessins TikZ, c'est long

→ But = rapidement **annoter ses slides** avec une tablette graphique via xournal++

→ et avec "duplices" on peut simuler des overlays !

et même ajouter des images !

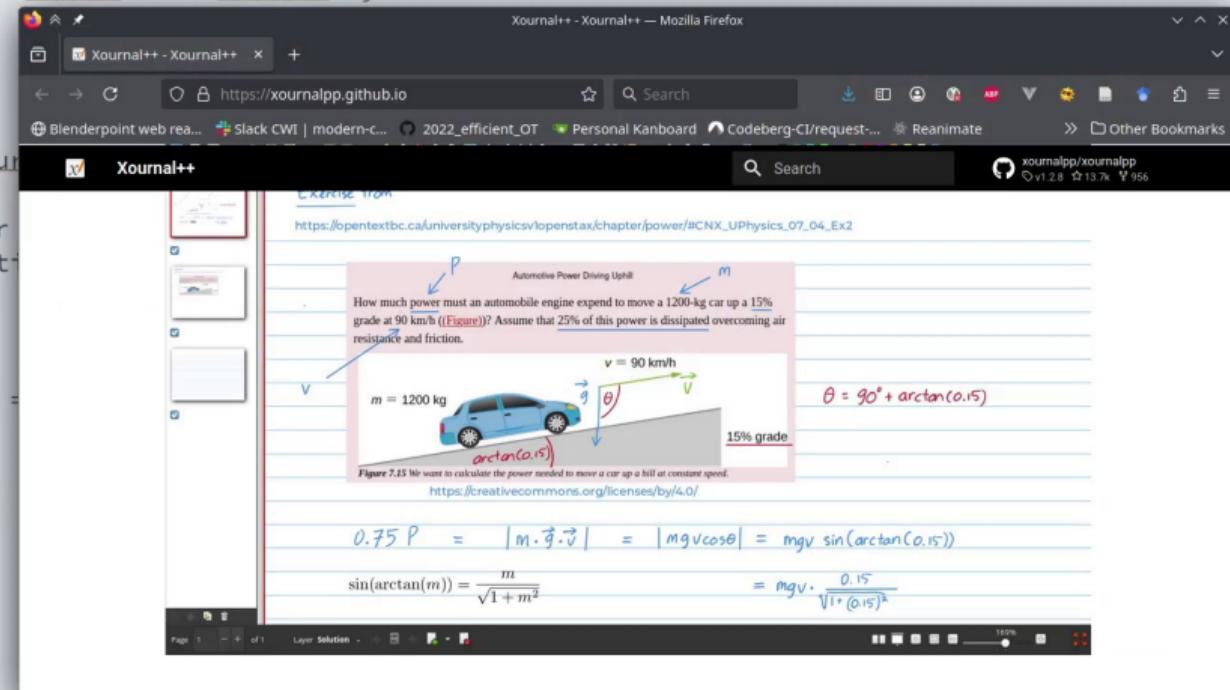


Léo Colisson | 1

269%

```
%% FRAME_START
\subsection{Annoter ses slides avec xournal++}
%% FRAME_STOP
```

```
%% FRAME_START
\begin{CacheMeCode}{xournal++}
\begin{frame}{Annoter ses slides avec xournal++}
\begin{importantB}{Motivation}
\bigskip
\$ \Rightarrow \$ But =
\end{frame}
\end{CacheMeCode}
%% FRAME_STOP
```

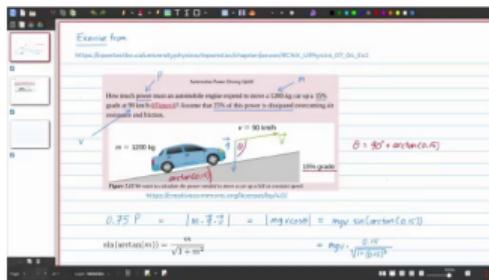


# Annoter ses slides avec xournal++

**Motivations** : faire des dessins TikZ, c'est long

→ But = rapidement **annoter ses slides** avec une tablette graphique via xournal++

→ et avec "dupliques" on peut simuler des overlays !



} et même ajouter des images !

```
%% FRAME_START
\subsection{Annoter ses slides avec xournal++}
%% FRAME_STOP

%% FRAME_START
\begin{CacheMeCode}{xournal=demo_slide,edit}

\begin{frame}{Annoter ses slides avec xournal++}
\importantB{Motivations} : faire des dessins  $\text{Ti}\text{\emph{k}}Z$ , c'est long

\bigs skip

\$ \Rightarrow \$ But = rapidement \importantB{annoter ses slides} avec une tablette graphique via xournal++
\end{frame}

\end{CacheMeCode}
%% FRAME_STOP
```

```
%% FRAME_START
\subsection{Annoter ses slides avec xournal++}
%% FRAME_STOP

%% FRAME_START
\begin{CacheMeCode}{xournal=demo_slide,edit}

\begin{frame}{Annoter ses slides avec xournal++}
\importantB{Motivations} : faire des dessins  $\text{Ti}\text{\emph{k}}Z$ , c'est long

\bigs skip

\$ \Rightarrow \$ But = rapidement \importantB{annoter ses slides} avec une tablette graphique via xournal++
\end{frame}

\end{CacheMeCode}
%% FRAME_STOP
```

Fichier Editer Affichage Navigation Journal Outils Greffon Aide

demo\_slide.xopp - Xournal++

Noto Sans Regular 30

Annotations:

- Motivations : faire des dessins TikZ, c'est long.
- ⇒ But = rapidement annoter ses slides avec une tablette graphique via xournal++

Annotations:

- Motivations : faire des dessins TikZ, c'est long.
- ⇒ But = rapidement annoter ses slides avec une tablette graphique via xournal++
- ⇒ En utilisant "duplicata", on peut simuler des overlays !

Annotations:

- Motivations : faire des dessins TikZ, c'est long.
- ⇒ But = rapidement annoter ses slides avec une tablette graphique via xournal++
- ⇒ En utilisant "duplicata", on peut simuler des overlays !
- et même ajouter des images !

Page 1 sur 3, Page PDF 1 Calque Calque 1 100%

Fichier Editer Affichage Navigation Journal Outils Greffon Aide

\*demo\_slide.xopp - Xournal++

Noto Sans Regular 30

1 2 3 4

Annoter ses slides avec xournal++

Motivations : faire des dessins TikZ, c'est long  
⇒ But = rapidement annoter ses slides avec une tablette graphique via xournal++

Annoter ses slides avec xournal++

Motivations : faire des dessins TikZ, c'est long  
⇒ But = rapidement annoter ses slides avec une tablette graphique via xournal++  
mais rigolo ☺

Annoter ses slides avec xournal++

Motivations : faire des dessins TikZ, c'est long  
⇒ But = rapidement annoter ses slides avec une tablette graphique via xournal++  
et avec "duplique" on peut simuler des overlays !

Page 1 sur 4, Page PDF 1 Calque Calque 1 100%

Fichier Editer Affichage Navigation Journal Outils Greffon Aide

demo\_slide.xopp - Xournal++

Noto Sans Regular 30

1 2 3 4

Annoter ses slides avec xournal++

Motivations : faire des dessins TikZ, c'est long  
⇒ But = rapidement annoter ses slides avec une tablette graphique via xournal++

Annoter ses slides avec xournal++

Motivations : faire des dessins TikZ, c'est long  
⇒ But = rapidement annoter ses slides avec une tablette graphique via xournal++

Annoter ses slides avec xournal++

Motivations : faire des dessins TikZ, c'est long  
⇒ But = rapidement annoter ses slides avec une tablette graphique via xournal++  
⇒ et avec "duplique" on peut suivre des overlays !

Page 1 sur 4, Page PDF 1 Calque Calque 1 Sauvegarder 100%

```
%% FRAME_START
\subsection{Annoter ses slides avec xournal++}
%% FRAME_STOP

%% FRAME_START
\begin{CacheMeCode}{xournal=demo_slide,edit} ■

\begin{frame}{Annoter ses slides avec xournal++}
\importantB{Motivations} : faire des dessins  $\text{Ti}\text{\texttt{emph}}{k}Z$ , c'est long

\bigs skip

\$ \Rightarrow \$ But = rapidement \importantB{annoter ses slides} avec une tablette graphique via xournal++
\end{frame}

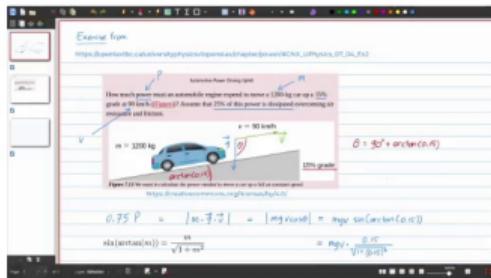
\end{CacheMeCode}
%% FRAME_STOP
```

# Annoter ses slides avec xournal++

**Motivations** : faire des dessins TikZ, c'est long

→ But = rapidement **annoter ses slides** avec une tablette graphique via xournal++

→ et avec "dupliques" on peut simuler des overlays !



et même ajouter des images !

```
%% FRAME_START
\subsection{Annoter ses slides avec xournal++}
%% FRAME_STOP

%% FRAME_START
\begin{CacheMeCode}{xournal=demo_slide,} →
\begin{frame}{Annoter ses slides avec xournal++}
\importantB{Motivations} : faire des dessins  $\text{Ti}\text{\emph{k}}Z$ , c'est long
\bigs skip
$ \Rightarrow $ But = rapidement \importantB{annoter ses slides} avec une tablette graphique via xournal++
\end{frame}
\end{CacheMeCode}
%% FRAME_STOP
```



Léo Colisson | 1

# Annoter ses slides avec xournal++

Motivations : faire des dessins TikZ, c'est long  
→ But = rapidement annoter ses slides avec une tablette graphique via xournal++

1

Motivations : faire des dessins TikZ, c'est long  
→ But = rapidement annoter ses slides avec une tablette graphique via xournal++

2

Motivations : faire des dessins TikZ, c'est long  
→ But = rapidement annoter ses slides avec une tablette graphique via xournal++  
To do now: "dupliquez" ou peut suivre des overlays !

3

Motivations : faire des dessins TikZ, c'est long  
→ But = rapidement annoter ses slides avec une tablette graphique via xournal++  
To do now: "dupliquez" ou peut suivre des overlays !

4

Motivations : faire des dessins TikZ, c'est long  
→ But = rapidement annoter ses slides avec une tablette graphique via xournal++  
To do now: "dupliquez" ou peut suivre des overlays !

I

mais rigolo

Smiley face icon

**Motivations : faire des dessins TikZ, c'est long**

⇒ But = rapidement **annoter ses slides** avec une tablette graphique via xournal++

↳ et avec "dupliques" on peut simuler des overlays !

et même ajouter des images !

Léo Colisson | 1

5 4 sur 4

# Aller plus loin

→ Mixer beamer et vidéo/3D/manim/...

# Classical Zero-Knowledge



# Classical Zero-Knowledge

Solution exists?

1			
			4
		3	
	2		



# Classical Zero-Knowledge



Yes!  
I won't reveal it.

Solution exists?

1			
			4
		3	
	2		



# Classical Zero-Knowledge



A low-poly 3D scene set in a red-toned landscape with jagged mountains in the background. In the foreground, a female character with long dark hair and a blue dress stands on the left, facing right. A male character with red skin and horns stands on the right, facing left. Between them is a 4x4 grid of 16 smaller squares. The squares are arranged in four rows and four columns. The numbers 1 through 4 are placed in specific squares: 1 is in the top-left, 4 is in the middle-right, 3 is in the bottom-middle, and 2 is in the bottom-left. The remaining squares are empty. Two speech bubbles are present: one from the female character saying "Yes! I won't reveal it." and one from the male character saying "I don't trust you." To the right of the male character is a glowing green circular device with several glowing blue spheres.

Yes!  
I won't reveal it.

I don't trust you.

# Classical Zero-Knowledge

Yes!  
I won't reveal it.

I don't trust you.

1			
		4	
		3	
	2		



# Classical Zero-Knowledge

Yes!  
I won't reveal it.

I don't trust you.

1	4	2	3
2	3	4	1
4	1	3	2
3	2	1	4



# Classical Zero-Knowledge



# Classical Zero-Knowledge



A low-poly 3D scene set in a red-hued landscape with jagged mountains. In the foreground, a character in a blue dress with red horns stands facing away from the camera. Another character in a red shirt and black pants with red horns stands facing the first character. Between them is a 4x4 grid of light blue rectangles. The top-left rectangle contains the number '1', the top-right '4', the middle-left '3', and the bottom-left '2'. The character in red is pointing towards the grid. Two speech bubbles are present: one from the character in blue saying 'Yes! I won't reveal it.' and one from the character in red saying 'I don't trust you.'

Yes!  
I won't reveal it.

I don't trust you.

# Classical Zero-Knowledge



A low-poly 3D scene set in a desert-like landscape with red mountains and small purple trees. Two characters are standing in front of a 4x4 grid of blue cards. The character on the left, wearing a blue dress and red horns, says "Yes! I won't reveal it." The character on the right, wearing a red shirt and red horns, says "I don't trust you." A glowing green structure with blue spheres is on the right.

Yes!  
I won't reveal it.

I don't trust you.

1			
	4		
		3	
	2		

# Classical Zero-Knowledge



# Classical Zero-Knowledge



# Classical Zero-Knowledge



# Classical Zero-Knowledge



Generalizable in a non-interactive way to NP problems.

# Mixer beamer et vidéo/3D/manim/...

On peut jouer la vidéo ci-dessus avec points d'arrêt + PDF généré automatiquement via :

```
\begin{CacheMe}{presetBlenderpointAddVideo,
    filename=Video_bank/Teaching/Zero_knowledge_proofs/zk_with_sudoku.mp4,
    nb frames=539, % Error if this change
    stops={0, 6, 10, 13, 60, 143, 222, 307, 425, 510, 518, 524, 537},
    add stops={284}, % This adds stops in the pdf only
    remove stops={307}, % This removes stops in the pdf only
    % Default speed is x2, except x3 from frame 0 to 3, and
    % remove frames 11-325:
    speed={2,0-50:3,119-325:0},
}
\end{CacheMe}
```

# Comment ça fonctionne ?

- **Compilation** : robust-externalize va extraire avec un script python les frames à garder pour le PDF
- **Inclusion** : robust-externalize inclut les frames nécessaires dans le PDF, et ajoute des annotations sur les frames à supprimer dans la vidéo et à remplacer par une vidéo avec des pauses.  
→ **PDF est natif !**
- **Post-commandes** : on lance  
`blenderpoint_produce_final_video.py presentation_robust_externalize.pdf`
  - qui lit le PDF,
  - qui transforme les slides en images (3 frames),
  - qui supprime les slides annotées et les remplace par des vidéos,
  - qui calcule le numéro des frames des points d'arrêts,
  - et les inclut dans les metadata de la vidéo.
- Utilise un **lecteur fait maison** qui joue la vidéo avec points d'arrêts  
<https://leo-colisson.github.io/blenderpoint-web/>

Divers

# Benchmark

Mesures de rapidité (pas refait récemment) :

- **sans externalisation : 1mn25**
- avec externalisation, premier lancement (non parallèle) : 3mn05 (2x plus lent)
- avec externalisation, premier lancement (parallèle) : 60s (1.5x plus rapide!)
- avec externalisation, lancements suivants (non compilé) : 5.7s (15x plus rapide!)
- avec externalisation, lancements suivants (compilé) : **4.0s (22x plus rapide!)**

## Autres fonctionnalités

- copy file to cache
- Renvoyer des données dans le document principal (e.g. compteur d'équation, taille...)
- Compilation de templates pour encore plus de vitesse
- Images matplotlib
- Opérations pour créer/modifier/évaluer/... les placeholders (e.g. à partir d'un fichier)
- Code bash
- Système d'import pour être plus efficace
- Remplacement partiel de placeholders
- Compilation manuelle (sans shell escape) et/ou en parallèle
- Commandes pour nettoyer le cache
- Forward automatique de couleurs
- Support inclu de gnuplot/bash/python/images en ligne/tikzit
- ...

# Limitations et TODO

# Limitations et TODO

- Liens non supportés (possiblement dans le futur, pour l'instant désactiver le cache pour les images problématiques)

# Limitations et TODO

- Liens non supportés (possiblement dans le futur, pour l'instant désactiver le cache pour les images problématiques)
- `remember picture` non supporté

# Limitations et TODO

- Liens non supportés (possiblement dans le futur, pour l'instant désactiver le cache pour les images problématiques)
- `remember picture` non supporté
- **arXiv** supprime automatiquement les .pdf, donc faire soit:
  - `python robExt-prepare-for-arxiv.py + \robExtConfigure{rename backup files for arxiv}`
  - ou `\robExtConfigure{backup source for arxiv}`avant de l'envoyer en ligne (ou désactiver cache pour arXiv).

# Limitations et TODO

- Liens non supportés (possiblement dans le futur, pour l'instant désactiver le cache pour les images problématiques)
- `remember picture` non supporté
- **arXiv** supprime automatiquement les .pdf, donc faire soit:
  - `python robExt-prepare-for-arxiv.py + \robExtConfigure{rename backup files for arxiv}`
  - ou `\robExtConfigure{backup source for arxiv}`avant de l'envoyer en ligne (ou désactiver cache pour arXiv).
- TODO: vérifier si les polices intégrées dans les sous-PDF **augmentent la taille du pdf**  
→ Toujours possible de désactiver `robust-externalize` pour le document final

# Limitations et TODO

- Liens non supportés (possiblement dans le futur, pour l'instant désactiver le cache pour les images problématiques)
- `remember picture` non supporté
- **arXiv** supprime automatiquement les .pdf, donc faire soit:
  - `python robExt-prepare-for-arxiv.py + \robExtConfigure{rename backup files for arxiv}`
  - ou `\robExtConfigure{backup source for arxiv}`avant de l'envoyer en ligne (ou désactiver cache pour arXiv).
- TODO: vérifier si les polices intégrées dans les sous-PDF **augmentent la taille du pdf**  
→ Toujours possible de désactiver robust-externalize pour le document final
- Booster encore plus la vitesse de la première compilation



Thankyou!