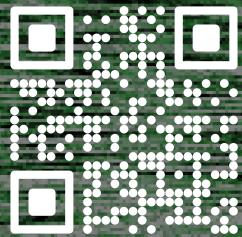


Using the Slixte class

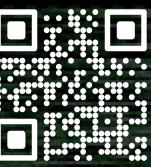
a step-by-step guide



Benoît Corsini

<https://ctan.org/pkg/slixte>

Table of contents







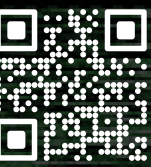




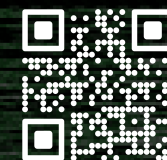
-  The Slixte class
-  Environments and commands
-  Customizing the slides
-  Going further

Table of contents



-  The Slixte class
-  Environments and commands
-  Customizing the slides
-  Going further

Importing the class



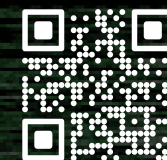
To download the **Slixte** class, click on the link below or scan the QR code at the top.

<https://ctan.org/pkg/slixte>

Once the class is downloaded, place the file `slixte.cls` in the same folder as the `.tex` file you will use for the slides (for example placing it in the main **Overleaf** project).

You are now ready to use the **Slixte** class! Get started either by using the file `template.tex` provided with the class, or by replacing the first line of your own file with

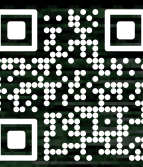
```
\documentclass{slixte}
```



By simply replacing the original class with the `Slixte` class and compiling the document, several errors likely arise.

- There might be unrecognized commands in your main document. It might be easier to remove everything at the start.
- If the main document is empty, an error will arise. To alleviate this issue, simply write the command `\titleslide` in your main document.
- By default the `Slixte` class requires two image files: `background.png` and `logo.png`.
 - You can either use your own images, or use the default ones from the package.
 - We will see in the section `Customizing the slides` how to change the default files or completely remove these images.

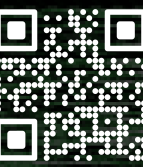
Basic structure of the class



The `Slixte` class creates slides, each occupying a full page. It is mostly powered by the `Beamer` class and the `TikZ` package. There are several ways to create a new slide.

- Using predefined full-page commands.
 - Title page: `\titleslide`, `\titlepage`, or `\maketitle`.
 - Table of contents: `\tableofcontents` or `\toc`.
 - Final page: `\finalslide`, `\finalpage`, or `\makefinal` (all taking an optional argument for the text to appear, and using the color `final` for the central text).
- Using one of the two slide environments.
 - Generic slide: `\begin{slide}...\end{slide}`.
 - Tikzslide: `\begin{tikzslide}...\end{tikzslide}`.

The slide environment



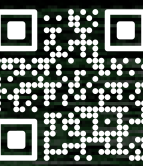
The `slide` environment created by calling `\begin{slide}...\end{slide}` creates a new slide taking one page.

- Whatever is placed between the `\begin{slide}` and `\end{slide}` is standard text.
- It is possible to set a title to the slide (otherwise empty) by using an optional argument within square brackets: `\begin{slide}[Title]...\end{slide}`.

Try adding the following code within your main document.

```
\begin{slide}[A slide]
  The text within the slide.
\end{slide}
```

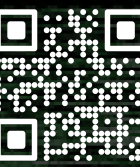
The tikzslide environment



The `tikzslide` environment created by calling `\begin{tikzslide}...\end{tikzslide}` creates a new slide taking one page.

- Whatever is placed between the `\begin{tikzslide}` and `\end{tikzslide}` belongs to a `tikzpicture` environment as provided by the `TikZ` package.
- The position of the origin can be modified (see [Customizing the slides](#) for more details).
- It is possible to set a title to the slide (otherwise empty) by using an optional argument within square brackets: `\begin{tikzslide}[title]...\end{tikzslide}`.
- Using the `\content{...}` command within the `tikzslide` environment places text as if it was part of a standard `slide` environment.

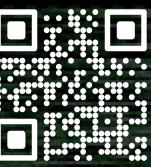
The tikzslide environment (example)







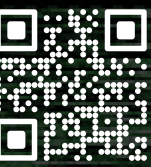
Try adding the following code within your main document.

```
\begin{tikzslide}[A tikzslide]
  \foreach \n in {1,...,9}{
    \node[draw, circle, first] at (\n, \n){};
  }
  \content{
    Some text inside the tikzslide.
  }
\end{tikzslide}
```

Table of contents

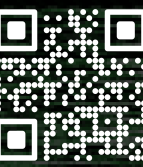


-  The Slixte class
-  Environments and commands
-  Customizing the slides
-  Going further



This section covers the following (clickable) topics.

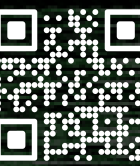
- [Global inputs.](#)
- [Sections.](#)
- [Basic commands and environments.](#)
- [Framed environments.](#)
- [References.](#)
- [Moving around the document.](#)
- [Final useful commands.](#)
- [Exercises.](#)



The `Slixte` class contains 4 global inputs. These should preferably be set in the preamble, but can also be changed within the main part of the document without creating errors.

- The title of the slides, modified by calling `\title{...}`. By default the title appears at the bottom left part of any slide (this can be changed, see [Customizing the slides](#)).
- The subtitle of the slides, modified by calling `\subtitle{...}`. This parameter is only used for the title page of the document.
- The author of the slides, modified by calling `\author{...}`. By default the author appears at the bottom right part of any slide (to change, see [Customizing the slides](#)).
- Extra info for the slides, modified by calling `\info{...}`. This parameter is only used for the title page of the document.

Global inputs (example)

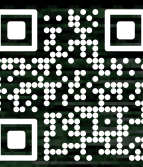


To re-create the title page of this document, include the following in the preamble.

```
\title{Using the Slixte class}
\subtitle{a step-by-step guide}
\author{Beno{\^i}t Corsini}
\info{\url{https://ctan.org/pkg/slixte}}
```

You can also check that this can be changed dynamically within the document.

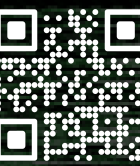
```
\titleslide
\author{Someone Else}
\titleslide
```



The `Slixte` class lets you organize the document in sections, although with a slightly different structure than usual.

- The preamble should include all the sections of the document, created via the command `\addsection[symbol]{section name}`. The `symbol` argument is optional and set to `§\bullet§` by default.
- Then, within the main part of the document, a new section is specified by calling the function `\section` without any input.
 - The section will naturally be used in the order they were created in the preamble.
 - If the command `\section` is called more times than `\addsection`, no compilation error will arise but the intermediate table of contents will be empty and the section name at the bottom of the page will not be updated.

Sections (example)



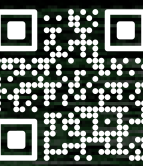
Try adding the following code within your preamble.

```
\addsection{A standard section}  
\addsection[$\times$]{A section symbolized by a cross}
```

And add this code within your main document.

```
\section  
\begin{slide}[A standard slide]\end{slide}  
\section  
\begin{slide}[A slide from the cross section]\end{slide}  
\section  
\begin{slide}[An unclear slide]\end{slide}
```

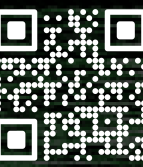
Basic commands and environments



The `Slixte` class contains various useful commands and environments.

- Item commands to standardize the style within each `itemize` environment.
 - `\citem[color]{item}` which creates custom colored items.
 - `\fillitem[color]` which creates colored and filled dots; equivalent to using `\citem[color]{ \bullet }`.
 - `\emptyitem[color]` which creates colored and empty dots; equivalent to using `\citem[color]{ \circ }`.
- Other environments useful for transitions.
 - `alert`, `remark`, `pointer`, and `question` environments, all with a single optional argument further explained on the next slide.

Basic commands and environments



⚠ This is the default `alert` environment. The alert symbol can be modified by entering an optional argument to the environment.

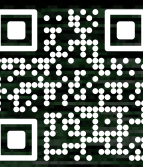
This environment is the fundamental one used to create all subsequent ones.

Note: This is the default `remark` environment. The “Note” text can be modified by entering an optional argument to the environment.

→ This is the default `pointer` environment. The color of the arrow can be modified by entering an optional argument to the environment.

Q: This is the default `question` environment. The color of the text can be modified by entering an optional argument to the environment.

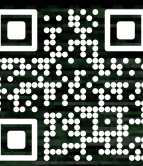
Basic commands and environments (example)



Try putting the following code within a `slide` environment.

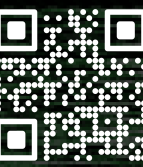
```
\begin{itemize}
  \fillitem A filled bullet.
  \emptyitem[second] An empty circle.
  \citem[third]{3.} A custom item.
\end{itemize}

\begin{question}A question worth asking?\end{question}
\begin{remark}[Answer]
  A remark that is actually an answer.
\end{remark}
```



As it is intended for scientific presentations, the `Slixte` class contains a few pre-implemented framed environments to highlight results.

- The fundamental `result` environment, taking two arguments, and applied by calling `\begin{result}{...}{...}...\end{result}`.
 - The first parameter is the desired name of the type of result.
 - The second parameter is the information on the result; when not empty, it will appear between brackets.
- The `theorem`, `definition`, and `problem` environments.
 - All three are based on the `result` environment.
 - They take one optional argument corresponding to the info of the result.



Definition

This is a generic definition, without any optional parameter specified.

Definition (More specific)

This is a more specific definition, with extra information specified.

Framed environments (example)



Try putting the following code within a `slide` environment.

```
\begin{result}{Quotation}{}  
    ‘‘You can create your own result box.’’ – Beno{\^i}t  
\end{result}
```

```
\begin{result}{Quotation}{Beno{\^i}t, Now}  
    You can create your own result box.  
\end{result}
```

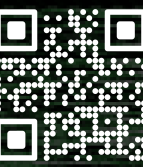
Framed environments (example)



Try putting the following code within a `slide` environment.

```
\begin{definition}
  A general definition that everyone should know.
\end{definition}
```

```
\begin{definition}[Math]
  A math definition that some people should know.
\end{definition}
```



On top of framed results, the `Slixte` class supports references.

- A new reference is created by calling the `\addref` command, which takes five arguments.
 - While it is not mandatory to follow this order, the arguments are intended to be: the name of the reference, the author(s), the year, the title, and the journal.
 - The name of the reference, corresponding to the first input of the `\addref` command, will appear as a superscript within the document.
- The references created can then be called anywhere in the document by using the `\citeref{refname}` command.
- The reference page can be shown anywhere in the document using the `\refslide`, `\refpage`, or `\makeref` command. All three commands take an optional input as the title of the reference slide.

References (example)



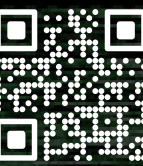
Try adding the following code within your preamble.

```
\addref{Y}{You}{yesterday}{Math?}{online}  
\addref{Me}{Me}{today}{Math!}{in-class}
```

And add this code within your main document.

```
\begin{slide}  
    You said something yesterday\citeref{Y},  
    and I only replied now\citeref{Me}.  
\end{slide}  
\refslide
```

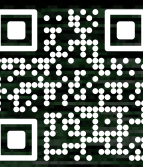

Moving around the document



To simplify the use of the resulting file, both when writing and presenting the slides, the **Slixte** class includes a few automated links.

- Clicking on the title (by default at the bottom left) brings you back to the title page.
- Clicking on the author (by default at the bottom right) brings you to the final page.
- Clicking on the section name (by default at the bottom and in the middle) brings you to the beginning of the section. This also applies to any section from the table of contents.
- Clicking on a reference (from the document) brings you to the reference page.
- Clicking on a reference (from the reference page) brings you to the first page where this reference is used.

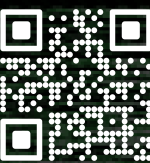
Note: When the clicking target does not exist, you are brought to the first page.



On top of the previous fundamental commands and environments, and before moving to more advanced methods, the `Slixte` class also implements various useful commands.

- `\spacing[length]` which creates vertical space (in cm, by default `0.5`).
- `\tosection{}` which creates direct links to sections (colored using `tosection`).
- `\break`, equivalent to the `\pause` command for `Beamer`.
- `\showon[window]{...}`, equivalent to `\onslide<window>{...}` for `Beamer`.
- `\makeon[window]{...}`, equivalent to `\only<window>{...}` for `Beamer`.
- `\spliton` and `\splitoff`, which switches on/off the previous three commands (useful for removing slide splits when creating the slides).

Final useful commands (example)



Try putting the following code within a `slide` in various sections.

You can directly go to `\tosection{1}` or `\tosection{2}`.

```
\spacing
```

You can also try to go to `\tosection{0}` or `\tosection{10}`.

```
\spacing[2]
```

Finally, we can also go to `\tosection{}`.

Final useful commands (example)



Try putting the following code within a `slide` environment.

```
A text on every slide.
```

```
\break
```

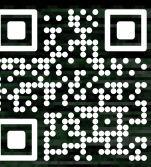
```
Some text appearing on the second slide.
```

```
\showon[3]{A line only shown on the third slide.}
```

```
\makeon[4]{A line only made on the fourth slide.}
```

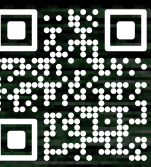
```
A final line differentiating show and make.
```

Exercise: create the following slide (1/3)



In this exercise, we are trying to create:

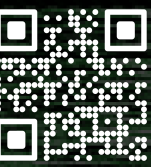
Exercise: create the following slide (1/3)



In this exercise, we are trying to create:

- A list of items.

Exercise: create the following slide (1/3)



In this exercise, we are trying to create:

- A list of items.
- With subtasks, such as:

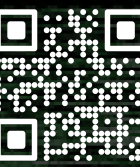
Exercise: create the following slide (1/3)



In this exercise, we are trying to create:

- A list of items.
- With subtasks, such as:
 - having different types of items;
 - with different colors.

Exercise: create the following slide (1/3)



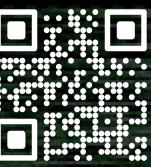
In this exercise, we are trying to create:

- A list of items.
- With subtasks, such as:
 - having different types of items;
 - with different colors.

Observation

If you can replicate everything on this slide, proceed to the next one.

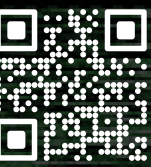
Exercise: create the following slide (2/3)



Open problem (Me, Now)

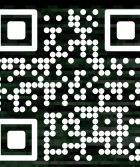
Can you still create what is in this slide??

Exercise: create the following slide (2/3)



→ The solution should be visible [here](#).

Exercise: create the following slide (2/3)

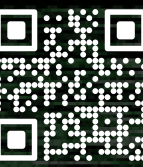


Open problem (Me, Now)

Can you still create what is in this slide??

Where to: You can also go to the beginning of the current section, also called **Environments and commands**.

Exercise: create the following slide (2/3)



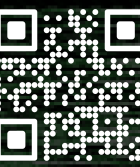
Open problem (Me, Now)

Can you still create what is in this slide??

→ The solution should be visible [here](#).

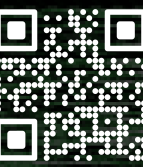
Where to: You can also go to the beginning of the current section, also called **Environments and commands**.

Exercise: create the following slide (3/3)



- Slixte (2000). **Back in the day.** *CTAN*.¹
- Slixte (Now). **Question.** *CTAN*.[?]
- Slixte (2025). **Reference page.** *CTAN*.^{here}
- Slixte (2030). **But also the future.** *CTAN*.⁴

Solution (1/3)



```
\begin{slide}[Exercise: create the following slide (1/3)]
  In this exercise, we are trying to create:
  \begin{itemize}\break
    \fillitem A list of items.\break
    \fillitem With suptasks, such as:\break
    \begin{itemize}
      \emptyitemhaving different types of items;
      \emptyitem[second] with different colors.
    \end{itemize}
  \end{itemize}\break

  \spacing

  \begin{result}{Observation}{}
    If you can replicate everything on this slide,
    proceed to the next one.
    \spacing[1]
  \end{result}
\end{slide}
```

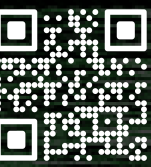
Solution (2/3)



```
\addref{1}{Slixte}{2000}{Back in the day}{CTAN}
\addref{?}{Slixte}{Now}{Question}{CTAN}
\addref{here}{Slixte}{2025}{Reference page}{CTAN}
\addref{4}{Slixte}{2030}{But also the future}{CTAN}

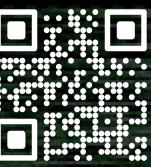
\begin{slide}[Exercise: create the following slide (2/3)]
  \showon[1,3-]{
    \begin{problem}[Me, Now]
      Can you still create what is in this slide?\citeref{?}
    \end{problem}}
  \makeon[2,4]{
    \spacing
    \begin{pointer}[third]
      The solution should be visible\citeref{here}.
    \end{pointer}}
  \showon[3-]{
    \spacing[2]
    \begin{remark}[Where to]
      You can also go to the beginning of the current section,
      also called \tosection{}.
    \end{remark}}
\end{slide}
```






Solution (3/3)

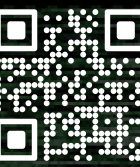


```
\refslide[Exercise: create the following slide (3/3)]
```

Table of contents



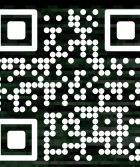
-  The Slixte class
-  Environments and commands
-  Customizing the slides
-  Going further



This section covers the following (clickable) topics.

- [Attributes of the class.](#)
- [Changing the design.](#)
- [Customization structure of the class.](#)
- [Changing the title page.](#)
- [Changing the banners.](#)
- [Changing the table of contents.](#)
- [Changing the content of the slides.](#)
- [Possible template.](#)
- [Summary of elements and attributes.](#)

Attributes of the class



The `Slixte` class has several optional attributes.

- By default, the page is 20cm high, but can be 10cm with `short` or 100cm with `tall`.
- The page ratio can be 1/1 with `square`, 3/2 with `photo`, 4/3 with `standard`, or 16/9 with `widescreen` (default).
- By default, the origin is at the bottom left of the page, but can be centered with `center`, or the top left of the page (with y going down) with `upper`.
- It is possible to cover pages with a grid to simplify the placement of items using `grid`.
- It is possible to show some of the boxes of the nodes on the page by using `info`.

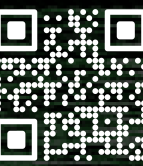
Attributes of the class (example)



Try the following code along with some of the other parameters.

```
\documentclass [short, upper, grid, info] {slixte}

\begin{document}
  \begin{tikzslide}
    \foreach \n in {1,...,9}{
      \node[draw, circle, first] at (\n, \n){};
    }
  \end{tikzslide}
\end{document}
```

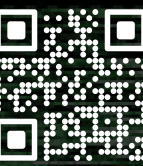


The two images used for the slides (`background` and `logo`) can be customized as follows

- Use the commands `\background{...}` and `\logo{...}` to change the files.
- Use the commands `\showbackground`, `\hidebackground`, `\showlogo`, and `\hidelogo` to either show or hide the background and logo.

It is also possible to customize the color scheme.

- There are five default colors used for the slides: `text` (black), `page` (white), `first` (green), `second` (purple), and `third` (red).
- These colors can directly be changed using `\definecolor` or `\colorlet`.
- Other color parameters will automatically be used when defined: `titlepage`, `author`, `slideheader`, `section`, `ref`, etc. A complete summary of these parameters can be found at the `end of the section`.



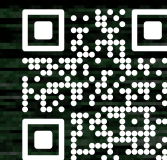
On top of the files and whether they appear or not, the `background` and `logo` can also be moved around. We focus here on the `background` parameters, but all the commands below can also be applied to the `logo`.

- `\backgroundx` and `\backgroundy`: the position on the page of the `background`.
- `\backgroundwidth` and `\backgroundheight`: the size of the `background`.
- `\backgroundanchor`: the `TikZ` anchor of the `background`.

Note: By default, only `\backgroundanchor` is defined and set to `center`. The other four parameters are automatically adapted to fit the structure of the slides (`short`, `tall`, `upper`, etc).

The other four parameters can be defined using `\newlength` or directly with `\def`.

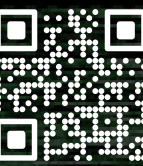
Changing the design (example)



Try the following code in your document.

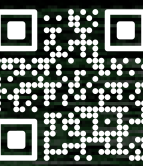
```
\logo{background.png}
\definecolor{titlepage}{RGB}{0, 100, 255}
\colorlet{author}{first}
\titleslide
\hidelogo
\def\backgroundanchor{south west}
\def\backgroundwidth{2cm}
\def\backgroundheight{0.5\paperheight}
\titleslide
```


General structure of the class



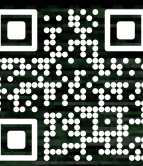
The *Slixte* class employs a few structural principles for its objects and commands.

- Each *element* of the class has a given name (`background`, `logo`, etc).
 - A lot of them have an *innate* command (`\background{...}`, `\title{...}`, etc).
 - Each of them has a set of *attributes*. These can be *pre-* or *post-attributes*.
 - *pre-attributes* are predefined commands (`\show`, `\hide`, etc).
 - *post-attributes* are parameters to customize (`x`, `width`, etc).
 - They may also have *relative attributes* (or *relatives*), such as colors, booleans, counters, etc. These usually share the same name as the *element*.
- For example, the *element* `background` has an *innate* command (changing the background file), two *pre-attributes* (`\show` and `\hide`), five *post-attributes* (`x`, `y`, `width`, `height`, and `anchor`), and a boolean *relative* (used to show or hide the background).



One *element* of the class is `titlepage`, with the following *attributes*.

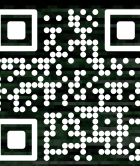
- One *innate* command with no input, creating the title page (see [Environments and commands](#)).
- One *pre-attribute*, `\make`, an alias for `\titlepage` and `\titleslide`.
- One *post-attribute*, `opacity`, corresponding to the opacity of the overlay on the image from `background`.
- One *relative*, corresponding to the color of the overlay on the image from `background`; this is the same as `text` by default.
 - On top of the `titlepage` color, there is a second *relative* color called `titlepagetext`, equal to `page` by default.



On top of `titlepage`, `background`, and `logo`, the title page is composed of four other *elements*: `title`, `subtitle`, `author`, and `info`. Their *innate* functions were already introduced in `Environments and commands` and their *attributes* are defined below.

- `\fontsize`: applies the defined font size of the *element*.
- `\setfontsize`: modifies the font size of the *element* using two inputs, one optional and one compulsory; the compulsory one is the font height and the optional one is the multiplier of the font spacing (by default 1.2).
- `x` and `y`: the position of the *element*.
- `anchor` and `align`: the corresponding parameters of the `TikZ` node.
- `width`: the text width parameter of the `TikZ` node.
- A color *relative*, equal to `titlepagetext` by default.

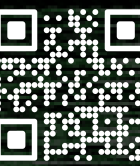
Changing the title page (example)



Try the following code in your document.

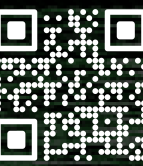
```
\def\titlepageopacity{1}  
\colorlet{titlepagetext}{third}  
\setfontsize{subtitle}{2.5cm}  
\def\titlex{0cm}  
\def\titley{\paperheight}  
\def\titleanchor{north west}  
\def\authorwidth{\paperwidth}  
\def\authoralign{left}  
\titleslide
```

Changing the banners



The two banners (top and bottom) on every slide are composed of a variety of *elements* explored in the next two slides. The four main ones are `banner`, `topbanner`, `bottombanner`, and `progress` (for the progress line at the bottom of the page). None of these *elements* have an *innate* command, but they share the following *attributes*.

- `\show`, `\hide`, `height`, and `opacity`.
- `\fontsize`, `\setfontsize`, `x`, `y`, `width` (only `topbanner` and `bottombanner`).
 - All these five *attributes* relate to the text within the banner.
 - The text of the banner is placed according to `x` and `y`, spanning `width` horizontally.
- `align` (only `progress`): either `left` (default) or `right`.
- A color *relative* (`text` by default, except `third` for `progress`).



On top of the more structural banner *elements*, there are six text elements from the banners: `bannertext`, `bannerheader`, `bannertitle`, `bannerauthor`, `bannersection`, and `bannernumber`. They have the following *attributes*.

- `\show`, `\hide`: by default, they are all visible except `bannernumber`.
 - `align`, with four options: `left` (`bannerheader` and `bannertitle` default), `center` (`bannersection` default), `right` (`bannerauthor` and `bannernumber` default), or a number (for example `0` is the same as `left`, and `1` the same as `right`).
 - A color *relative* (`page` by default).
- ⚠ The `bannertitle` refers to the *element* of the banners related to `title` (for the whole slide, for example “Using the Slixte class” here), and not what is at the top of the slide, instead referred to as the `bannerheader` (“Changing the banners” here).

Changing the banners (example)



Try the following code in your document.

```
\def\banneropacity{1}
\def\topbannerheight{0.5\paperheight}
\setfontsize\topbanner{2cm}
\colorlet{bottombanner}{second}
\def\bottombannery{1cm}
\def\bannerheaderalign{right}
\showbannernumber
\def\bannernumberalign{0.75}
\begin{slide}[The bannerheader element]\end{slide}
```

Changing the table of contents

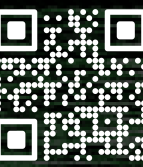


The table of contents part of the slides has three *elements*: `toc`, `section`, and `symbol`. The *innate* and all *pre-attributes* of `toc` also work with the alias `tableofcontents`.

Due to the diversity of roles of the *elements*, we cover their *attributes* separately, starting with `toc`. This *element* covers properties of the main table of contents and the repeated ones for each section.

- The *innate* was covered in [Environments and commands](#). It takes an optional input for the header of the slide, which then applies to all subsequent table of contents.
- `\show` and `\hide`: whether each section shows the current table of contents or not.
- `y` and `height`: vertical position and range of the sections within the table of contents.
- `opacity`: the opacity of the non-current sections in the table of contents.

Changing the table of contents

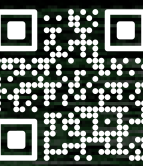


The *element section* covers properties of the overall organization into sections and the design of the section names in the table of contents. The *attributes* related to the organization were covered in *Environments and commands* (`\add`, `\to`, and the *innate*).

On top of these *attributes* already defined, every other one of `section` is shared with `symbol` and described below.

- `\show`, `\hide`, `\fontsize` and `\setfontsize`.
- `x`: the horizontal position of the `section` or `symbol`.
- `anchor`: the anchor defined for *TikZ* nodes.
- A color *relative* (by default `text` for `section` and `first` for `symbol`).

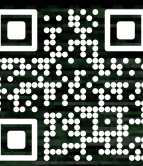
Changing the table of contents (example)



Try the following code in your document.

```
\def\tocheight{2cm}  
\def\sectionx{0.5\paperwidth}  
\def\sectionanchor{center}  
\setfontssize{symbol}{2cm}  
\toc[My Table of Contents]  
\hidesymbol  
\section  
\hidesection  
\toc
```

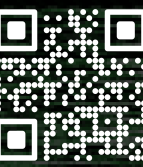
Changing the content of the slides



Within the main content part of the slides, there are four *elements*: `content`, `item`, `framing`, and `ref`. Since `framing` is the most complex, let us start with the other three.

- Attributes of `content`:
 - `\fontsize`, `\setfontsize`, and *innate* (see [Environments and commands](#)).
 - `align` (default `justify`), `opacity` (default `0.99`), and `margin` (default `1.8cm`).
- Attributes of `item`:
 - `\c`, `\fill`, `\empty` (see [Environments and commands](#)).
 - A color *relative* (by default `first`).
- Attributes of `ref`:
 - `\add`, `\cite`, and `\make` (see [Environments and commands](#)).
 - `\fontsize`, `\setfontsize` and a color *relative* (by default `first`).

Changing the content of the slides



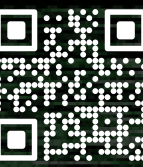
The final *element* of the main content, `framing`, relates to the framed environment (see [Environments and commands](#)) and has the following *attributes*.

- `align`: how the text should align within the environment (default `justify`).
- `width`: the total width of the framed environment.
- `space`: the space at the top of the environment (default `0.5cm`).
- `margin`: the margin between the frame and the inner text (default `1cm`).
- A color *relative* (by default `second`).

Unique to `framing` is the *sub-element* called `framingbox`, used for the actual structure of the frame and with the following *attributes*.

- `width`, `height`, `linewidth`, `roundedcorners`, and a color *relative*.

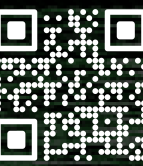
Changing the content of the slides (example)



Try the following code in your document.

```
\def\contentopacity{0.9}
\begin{slide}
  \begin{theorem}Some theorem.\end{theorem}
  \def\framingwidth{0.5\paperwidth}
  \def\framingmargin{0cm}
  \colorlet{framingbox}{third}
  \def\framingboxheight{-1cm}
  \begin{theorem}Some theorem.\end{theorem}
\end{slide}
```

Possible template: dark theme and bright colors



Title of the Slides
some subtitle
Author Name
extra info

Table of Contents

- Section 1
- Section 2
- Section 3
- Section 4

Title of the Slides Section 1 Author Name

A classical slide

Theorem
The roots of the degree two polynomial $ax^2 + bx + c$ are given by

$$\frac{b \pm \sqrt{b^2 - 4ac}}{2a}$$

This result is quite famous because:

- It is learned by a lot of students; and
- It looks more complicated than anything done before.

Title of the Slides Section 1 Author Name

```
\hidebackground
\hidelogo
\hidesymbol

\definecolor{first}{RGB}{65, 105, 225}
\definecolor{second}{RGB}{255, 215, 0}
\definecolor{third}{RGB}{220, 20, 60}
\definecolor{fourth}{RGB}{34, 139, 34}

\colorlet{page}{black}
\colorlet{text}{white}

\colorlet{titlepage}{page}
\colorlet{title}{first}
\colorlet{subtitle}{third}
\colorlet{author}{second}
\colorlet{info}{fourth}

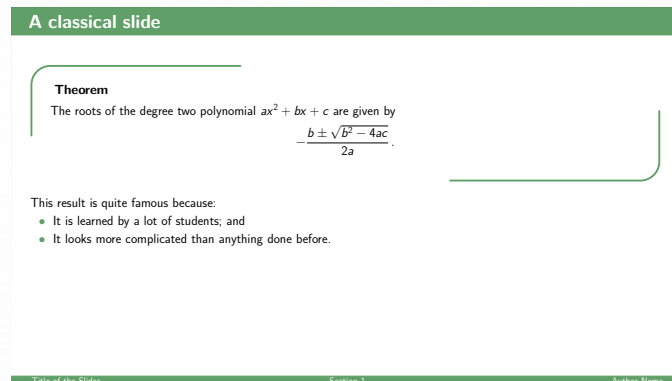
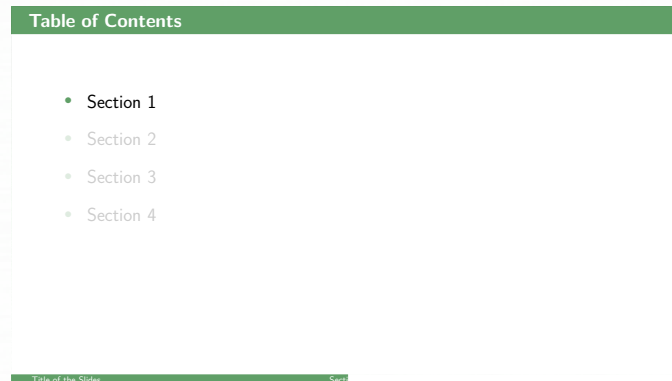
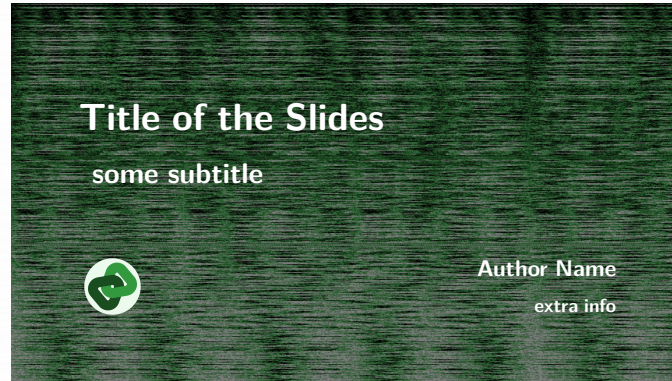
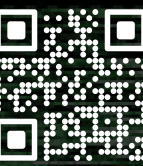
\colorlet{banner}{page}
\colorlet{bannerheader}{fourth}
\colorlet{bannertitle}{first}
\colorlet{bannerauthor}{second}
\colorlet{bannersection}{third}
\colorlet{progress}{fourth}

\colorlet{section}{third}
\colorlet{item}{third}
\colorlet{framingbox}{first}

\def\titlealign{center}
\def\subtitlealign{center}
\def\authoralign{center}
\def\infoalign{center}

\def\titlepageopacity{0.9}
\def\contentopacity{0.9}
\def\banneropacity{1}
\def\progressopacity{1}
```

Possible template: denser with overriding progress



```
\colorlet{topbanner}{first}  
\colorlet{bottombanner}{page}  
\colorlet{bannertext}{page}  
\colorlet{progress}{first}
```

```
\colorlet{framing}{text}  
\colorlet{framingbox}{first}
```

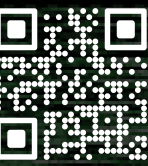
```
\setfontstoptopbanner{1}  
\def\banneropacity{1}  
\def\topbannerheight{1.5cm}  
\def\topbannerwidth{0.95\paperwidth}
```

```
\setfontsizebottombanner{0.5}  
\def\bottombannerheight{0.5cm}  
\def\progressheight{-1cm}  
\def\progressopacity{1}
```

```
\def\tocy{0.75\paperheight}  
\def\tocheight{0.3\paperheight}  
\def\sectionx{4cm}  
\def\symbolx{3cm}
```

```
\setfontsizecontent{0.7}  
\setlength{\contentmargin}{1cm}  
\def\contentopacity{1}
```

Possible template: grayscale with page number



Title of the Slides
some subtitle

Author Name
extra info

```
\hidebackground
\hidelogo
\hideprogress

\colorlet{first}{white!75!black}
\colorlet{second}{white!25!black}
\colorlet{third}{white!50!black}

\colorlet{titlepagetext}{black}
\colorlet{banner}{white}
\colorlet{bannertext}{black}
\colorlet{page}{black}
```

Table of Contents

- Section 1
- Section 2
- Section 3
- Section 4

Title of the Slides Author Name Section 1 2 / 3

```
\def\contentopacity{0.01}
\def\titlepageopacity{0.05}

\setfontsize{title}{2}
\def\subtitlex{0.1\paperwidth}
\def\subtitley{0.6\paperheight}
\setfontsize{subtitle}{1.8}
\def\authory{0.4\paperheight}
\def\authoralign{left}
\setfontsize{author}{2}
\def\infoy{0.3\paperheight}
\def\infoalign{left}
\setfontsize{info}{1.8}
```

A classical slide

Theorem
The roots of the degree two polynomial $ax^2 + bx + c$ are given by

$$\frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

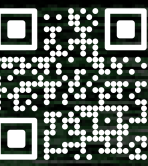
This result is quite famous because:

- It is learned by a lot of students; and
- It looks more complicated than anything done before.

Title of the Slides Author Name Section 1 3 / 3

```
\showbannernumber
\def\bannertitlealign{-0.32}
\def\bannerauthoralign{-0.115}
\def\bannersectionalign{left}
\def\bannersectionnumber{right}
\def\bottombannerx{0.65\paperwidth}
\def\bottombannerwidth{0.65\paperwidth}
```

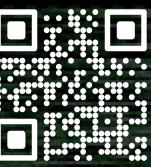

Summary of elements and attributes







<i>element</i>	author	background	banner	bannerauthor	bannerheader	bannernumber	bannersection	bannertext	bannertitle	bottombanner	content	framing	framingbox	info	item	logo	progress	ref	section	subtitle	symbol	tableofcontents	title	titlepage	titlepagetext	toc	topbanner
<i>innate</i>	X	X									X			X	X				X	X		X	X	X		X	
<i>\add</i>																		X	X								
<i>\c</i>															X												
<i>\cite</i>																		X									
<i>\empty</i>															X												
<i>\fill</i>															X												
<i>\fontsize</i>	X									X	X	X		X				X	X	X	X		X				X
<i>\hide</i>		X	X	X	X	X	X	X	X	X						X	X		X		X	X				X	X
<i>\make</i>																		X					X				
<i>\setfontsize</i>	X									X	X	X		X				X	X	X	X		X				X
<i>\show</i>		X	X	X	X	X	X	X	X	X						X	X		X		X	X				X	X
<i>\to</i>																			X								
<i>align</i>	X			X	X	X	X		X		X	X		X			X		X	X	X		X				
<i>anchor</i>	X	X												X		X			X	X	X		X				
<i>height</i>		X	X							X				X		X	X									X	X
<i>linewidth</i>													X														
<i>margin</i>											X	X															
<i>opacity</i>			X							X	X						X						X			X	X
<i>roundedcorners</i>													X														
<i>space</i>												X															
<i>width</i>	X	X										X	X	X	X					X			X				
<i>x</i>	X	X												X	X				X	X	X		X				
<i>y</i>	X	X												X	X					X			X			X	
<i>color relative</i>	X		X	X	X	X	X	X	X	X		X	X	X	X		X	X	X	X	X		X	X	X		X

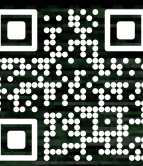
By default, the *attributes* `height`, `width`, `x`, `y` and the color *relative* are not defined.

Table of contents



-  The Slixte class
-  Environments and commands
-  Customizing the slides
-  **Going further**

Fundamental environment

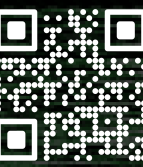


To understand the `Slixte` class further, it is worth mentioning the most fundamental environment: `page`. This environment does not take any argument, creates a single empty page, and is used for every subsequent environment and command. The result belongs to a `TikZ` environment and is otherwise empty (no background, banner, text box, etc).

The `page` environment also hosts two very useful commands: `\prepage` and `\postpage`. As their names entail, these commands are automatically called at the beginning and end of a `page` environment (within the `tikzpicture` environment themselves).

Finally, in order to create the different parts of the slide, there exists an *attribute* not mentioned before: `\draw`. This *pre-attribute* should only be used within the `TikZ` environment (so not within a standard `slide` for example).

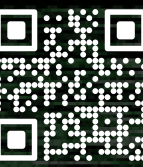
Fundamental environment (example)



Try the following code in your document.

```
\renewcommand{\backpage}{  
  \node[third, scale=5] at (15, 10){Background};  
}  
\renewcommand{\frontpage}{  
  \node[scale=5] at (15, 5){Foreground};  
  \node{\includegraphics[height=5cm]{background.png}};  
}  
\begin{page}\end{page}  
\begin{slide}\end{slide}
```

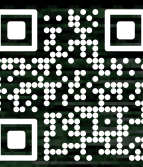
The draw attribute



Here is a non-exhaustive list of the *elements* with the `\draw attribute`. The possible inputs of the commands are directly written with the *element*.

- `background` and `overlay{opacity}{color}`.
- `banner`, `topbanner`, `bottombanner`, and `progress`.
- `top/bottombannertext [color]{align}{text}`.
- `bannerheader{header}` and `bannertitle/author/section/number` (with `bannertext` combining all four of them).
- `content`: creates the content background.
- `framingbox`: creates the framing box (using the node `framing`).
- `ref{refname}{author}{year}{title}{journal}`.

The draw attribute

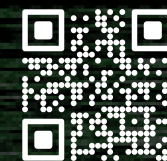


While most *elements* actually have the `\draw attribute`, the previous list highlights the ones relevant when further customizing the slides (the acute user of the `Slixte` class is invited to check within the `.cls` file for the other `\draw` methods).

The following slides now explain how these commands can be used to create custom slides, or how they can be redefined to fit the various needs.

Before that, the code behind the `slide` environment is provided, hoping it helps clarify the use of some of these commands.

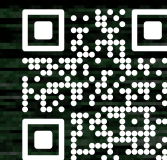
The draw attribute (example)



The following code will create a standard `slide` environment.

```
\begin{page}
  \drawbackground
  \drawbanner
  \drawcontent
  \drawprogress
  \drawbannerheader{Slide Header}
  \drawbannertext
  \content{The slide content.}
\end{page}
```

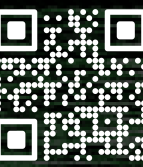
The draw attribute (example)



Try the following code in your document.

```
\begin{page}
  \drawbackground
  \drawoverlay{0.5}{red}
  \drawbannerheader{Slide Header}
  \showbannernumber
  \drawbannernumber
  \content{The slide content.}
\end{page}
```

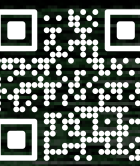

Customizing the commands



Some of the `\draw` commands can be re-written to fit personal tastes.

- `bannerheader/title/author/section/number`, for example to place these *elements* elsewhere. It is worth noting that these commands call the *attribute name* from their corresponding *element* (`\titlename`, `\authorname`, etc).
- `framingbox`: to draw the framing box and subsequent environments (`theorem`, `definition`, etc) differently.
- `ref`, to restructure the way references are written.

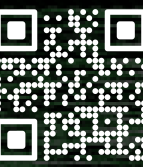
Customizing the commands (example)



Try the following code in your document.

```
\renewcommand{\drawbannerheader}[1]{  
  \node[first, scale=5]  
  at (\paperwidth/2, \paperheight/2){#1};  
}  
\renewcommand{\drawbannernumber}{  
  \drawtopbannertext [yellow]{right}  
  {page \insertpagenumber\ out of \insertdocumentendpage}  
}  
\begin{slide}[Title]Some slide\end{slide}
```

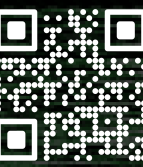
Customizing the commands (example)



Try the following code in your document.

```
\renewcommand{\drawframingbox}{  
  \draw[line width=0.1cm] (framing.north) -- (framing.south);  
  \draw[line width=0.1cm] (framing.west) -- (framing.east);  
  \draw[red, fill=red, opacity=0.2, line width=0cm]  
    (framing.north east) rectangle (framing.south west);  
}  
\begin{slide}  
  \begin{theorem}Some theorem.\end{theorem}  
\end{slide}
```

Customizing the commands (example)

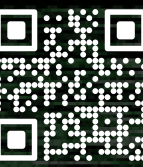


Try the following code in your document.

```
\renewcommand{\drawref}[5]{  
  \hyperlink{#1}{\textbf{#4}}  
  \hfill\textcolor{text!10!page}{#1}  
  \begin{itemize}  
    \emptyitem Written by #2 in #3.  
    \emptyitem Published in \textit{#5}.  
  \end{itemize}  
}
```

`\refslide`

Conclusion



I hope this document clarifies the use and options of the [Slixte](#) class.

If you still have questions/remarks, feel free to contact me at
benoitcorsini@outlook.com.

In particular, if the class does not work or behaves differently than expected, please contact me with your code and an explanation of what is wrong, so that I can try to fix the issue.

I hope you will enjoy using this slide template as much as I do!

Thank you!

Thank you!

Thank you!

Thank you!

Thank you!

Thank you!

Thank you!

Thank you!

Thank you!