TGM-HIT SEW Lecture

Typst template for lecture documents ("Skripten"), focused on software engineering

v0.1.0   December 02, 2025

https://github.com/TGM-HIT/typst-sew-lecture

Clemens Koza

Contents

I Introduction

This template is aimed at teachers of the information technology department at the TGM technical secondary school in Vienna, specifically those teaching software engineering. It can be used both in the Typst app and using the CLI:

Using the Typst web app, you can create a project by e.g. using the "Create new project in app" button on the package's Universe page: https://typst.app/universe/package/tgm-hit-sew-lecture.

To work locally, use the following command:

```bash
1 typst init @preview/tgm-hit-sew-lecture
```

To get started, you will likely be better off looking into the document created by the template: it contains instruction and examples on the most important features of this template. If you have not yet initialized the template, a rendered version is linked in the README, but it is recommended to view the source code along with the rendered form. If you are new to Typst, also check out the Typst documentation: https://typst.app/docs/.

The rest of this manual documents the individual functions offered by this package. This is useful if you want to know what customization options are available, or you're not sure what parts of the template package do.

As a school-specific template, this package is fairly opinionated and may not offer enough configurability to fit your needs. However, if you like this template, feel free to adapt the code (MIT-licensed) to your needs, or open a Github issue to request making this template more general.

II Module reference

II.a `tgm-hit-sew-lecture`

- `zebraw()`
- `no-zebraw()`
- `lines()`
- `template()`
- `colorbox()`
- `pinit-code-from()`
- `licenses`

`zebraw(..args: arguments, body: content) -> content`

A wrapper around `zebraw.zebraw()` that applies a few extra settings that would otherwise reset when calling `zebraw()` again (e.g. for setting the range of displayed lines)

```typ
1  ```typ
2  as
3  df
4  ```
5  Only line 2, same styles:
6  #{
7    show: zebraw.with(line-range:
       lines("2"))
8    ```typ
9    as
10   df
11   ```
12 }
```

```
1 as
2 df
```

Only line 2, same styles:

```
2 df
```

Parameters:

`..args` (`arguments`) – any custom zebraw arguments

`body` (`content`) – the content that should be styled

`no-zebraw(body:` `content` `) ->` `content`

Manually resets the raw code display settings configured by `zebraw()`.

```typ
1  ```typ
2  as
3  df
4  ```
5  Don't use styled code blocks:
6  #no-zebraw[
7    ```typ
8    as
9    df
10   ```
11 ]
```

```
1 as
2 df
```

Don't use styled code blocks:

```
as
df
```

Parameters:

`body` (`content`) – the content that should be styled

`lines(spec) ->` `array`

Helper function for specifying zebraw line numbers. It converts a string of the form `"2-4"` into the array `(2, 5)`: lower bound inclusive, upper bound exclusive line range.

Note the line numbers in this example:

```typ
1 #zebraw(line-range: lines("2-4"))[
2   ```typ
3   ...
4   ```
5 ]
```

```
2   b
3   c
4   d
```

```
template(
  license:  content ,
  header-left:  content   auto ,
  header-center:  content   auto ,
  header-right:  content   auto ,
  footer-left:  content   auto ,
  footer-center:  content   auto ,
  footer-right:  content   auto ,
  font:  string ,
) ->  function
```

The main template function. Your document will generally start with

```
1  #set document(..)  // title etc.                                          typ
2  #show: template(..)
```

which it already does after initializing the template.

Parameters:

license (content = none) – The license to show in the footer (by default)

header-left (content or auto = auto) – The content for the left header column; empty by default

header-center (content or auto = auto) – The content for the center header column; contains the title by default

header-right (content or auto = auto) – The content for the right header column; contains a date-based version number by default

footer-left (content or auto = auto) – The content for the left footer column; contains copyright information by default

footer-center (content or auto = auto) – The content for the center footer column; contains the page number and count by default

footer-right (content or auto = auto) – The content for the right footer column; empty by default

font (string = "Noto Sans") – The main font to use; note that even though this can be configured, some measurements, particularly those used by pinit-code-from(), may be wrong if you change it.

colorbox(body: content , color: color , ..args: arguments) -> dictionary

A colored box powered by showybox.

```
1  #grid(columns: 2,                            typ
2     colorbox[foo],
3     colorbox(color: red)[bar],
4  )
```

┌─────────────┐ ┌─────────────┐
│ foo   │ │ bar   │
└─────────────┘ └─────────────┘

Parameters:

body (content) – the content that should be displayed

color (color = green) – the color to base border and background on

`..args` (`arguments`) – any custom showybox arguments

```typ
pinit-code-from(
   color,
   pin,
   offset,
   body,
   looseness,
   width,
   ..args,
   bod,
) -> content
```

A wrapper around `pinit.pinit-point-from()`. It handles placing the note differently, based on a raster aligned to the raw code block. Pins inside the raw block are automatically set when a string of the form `PINn` (where `n` is a number) appears in the code, and `pinit-code-from()` can then refer to it.

```typ
1  ```typ
2  as      v- the note starts 6 monospace
3  dfPIN1              chars over by default
4     ^^^^^--- the arrow takes up 5 chars
5          by default
6  ```
7  #pinit-code-from(1)[here]
```

```
1 as      v- the note starts 6 monospace
2 df ←——— here      chars over by default
3    ^^^^^--- the arrow takes up 5 chars
4          by default
```

(The monospace grid alignment is more exact when using the template's fonts instead of the manuals'.)

```typ
1  ```typ
2   as
3   dfPIN2
4   gh
5  ```
6  #pinit-code-from(2,
7     pin: (0, 0, top+right),
8     offset: (5, -1, left),
9     width: 70%,
10 )[The anchor can be offset from the char
11    it's at. Automatic line breaking is
12    possible with `width`.]
```

```
1 as
2 df
3 gh
```

The anchor can be offset from the char it's at. Automatic line breaking is possible with `width`.

licenses: `dictionary`

A dictionary of links to various Creative Commons licenses. Each link is displayed using `ccicons`.

```typ
1 #licenses.cc-by-4-0 ---
2 #licenses.cc-by-nc-sa-4-0 ---
3 #licenses.cc-zero-1-0
```

cc BY — cc BY NC SA — cc 0