IEEE with Typst

Bob BOB
Bob Labs
Paris, France
bob.bob@bob-labs.org

Bob BOB
Bob Labs
Paris, France
bob.bob@bob-labs.org

Bob BOB
Bob Labs
Paris, France
bob.bob@bob-labs.org

***Abstract*— This document describes the most common article elements and how to use the TYPST to produce files that are suitable for submission to the Institute of Electrical and Electronics Engineers (IEEE).**

***Index Terms*—Class, IEEE, TYPST, paper, style, template, typsetting.**

I. Introduction

WELCOME to the Typst IEEE template. This template aims to provide the basis to create IEEE papers. Documentation and tutorials are available on the official Typst website: https://typst.app.

II. Why Typst ?

Typst is fast 🚀

Typst is open sourced 📖

Typst is easy ☑

More seriously:

- Previews your changes instantly.
- Provides clear, understandable error messages.
- Has a consistent styling system for configuring everything from fonts and margins to the look of headings and lists.
- Uses familiar programming constructs instead of hard-to-understand macros.

III. How to use Typst ?

A web playground is disponible on the official website, but it is also possible to use your favourite text editor thanks to the Typst lsp. Plugins exist for VSCode: https://github.com/Enter-tainer/typst-preview or even Vim: https://github.com/kaarmu/typst.vim.

IV. Where to get the Typst Templates

As always, the **Awesome GitHub Pages** are a good start. The Typst awesome page (https://github.com/qjcg/awesome-typst) already refers to articles about tools, integrations, templates, tutorials and much more!

V. Project Structure

A Typst project possess a main script, usually `main.typ`, ressources that can be managed inside folders, templates and a bibliography file. The structure may look as follows:

```
ressources/
├─ png/
│  ├─ img1.png
main.typ
template.typ
refs.bib
```

VI. Using the Template

The template is basically a function with arguments that has to be called at the start of the `main.typ` script.

A. Paper Title

The title of your paper is coded as:

```
title: "IEEE with Typst",
```

B. Author Names and Affiliations

The author section should be coded as follows:

```
authors: (
    (
      name: "Bob BOB",
      department: [President],
      organization: [Bob Labs],
      location: [Paris, France],
      email: "bob.bob@bob-labs.org"
    ),
    (
      name: "Bob BOB",
      department: [Vice president],
      organization: [Bob Labs],
      location: [Paris, France],
      email: "bob.bob@bob-labs.org"
    ),
    ...
)
```

C. Header and Footer

The header is present on each page whereas the footer is only printed in the first page.

```
header: "JOURNAL OF TYPST CLASS FILES, VOL. 18, NO.
```

979-8-3503-2934-6/23/\$31.00 ©2023 IEEE

```
9, SEPTEMBER 2020",
footer: "979-8-3503-2934-6/23/\$31.00 ©2023 IEEE",
```

D. Index terms and bibliography file

```
index-terms: ("Class", "IEEE", "TYPST", "paper",
"style", "template", "typsetting"),
bibliography-file: "refs.bib",
```

It might even be possible to add **keywords**:

```
keywords: [bananas, apples, oranges],
```

E. Abstracts

The coding is simply:

```
abstract: [
   This document describes the most common article
elements and how to use the TYPST to produce files
that are suitable for submission to the Institute of
Electrical and Electronics Engineers (IEEE).
],
```

F. Initial Drop Cap Letter

The first text paragraph uses a "drop cap" followed by the first word in ALL CAPS. This template uses the module `droplet`. It is imported at the beginning of the `template.typ` script with `#import "@preview/droplet:0.2.0": dropcap` and used as follows in the introduction section of `main.typ`:

```
#dropcap("WELCOME to the Typst IEEE template. This
template aims to provide the basis to create IEEE
papers. Documentation and tutorials are available on
the official Typst website: https://typst.app.")
```

Height and gaps can be modified in the template file or directly in the main file by slightly adding arguments to the function.

G. Modify everything at will

An important thing to take into account is that the whole code of the template is written in the `template.typ` file, so changing things is as simple as modifying a line in this file.

VII. Body

A. Sections and Subsections

Section headings are not as complicated as their LaTeX's counterparts. It is simply created with "=":

```
= Section Head
The text of your paragraph . . .

== Subsection
=== Subsubsection
...
```

B. Citations to the Bibliography

Coding for bibliography is included in the standard librairie. The `refs.bib` can be configured as follows:

```
@article{example,
  title={Example},
  author={Example, Example and Example, Example},
  journal={Example Example Example Example},
  volume={1},
  pages={1--2},
  year=2020,
  publisher={Example Example Inc.}
}
```

Then for a single citation code as follows:

```
see @bob
```

This will display as: see [1]
For multiple citations code as follows:

```
@bob1 @bob2 @bob3
```

This will display as [1]–[3]

C. Figures

Figures are coded with the standard Typst command as follows:

```
#figure(
  image("ressources/png/fig1.png"),
  caption: [This is the caption for one fig.]
)<fig1>
```

To cross-reference your figures in the text use the following code example:

```
See figure @fig1 ...
```

This will produce:
See figure 1 . . .

Fig1: This is the caption for one fig.

D. Tables

Tables should be coded with the standard Typst coding. The following example shows a simple table.

```
#figure(
  table(
  columns: (auto, auto, auto),
  inset: 3pt,
  align: horizon,
  [Order of filter], [Arbitrary coefficients $e_m$],
[coefficients $b_(i,j)$],
    [1],[$b_(i  j)=hat(e).  hat(beta_(i,j))$],[$b_(0
0)=0$],
  [2],[$beta_(2 2) = (1, -1, -1, 1, 1, 1)$],[],
   [3],[$b_(i  j)=hat(e).  hat(beta_(i  j))$],[$b_(0
0)=0$],
  ),

  caption: [A simple table example],
)<tab1>
```

Table1: A simple table example

Order of filter	Arbitrary coefficients e_m	coefficients $b_{i,j}$
1	$b_{ij} = \hat{e}.\widehat{\beta_{i,j}}$	$b_{00} = 0$
2	$\beta_{22} = (1, -1, -1, 1, 1, 1)$	
3	$b_{ij} = \hat{e}.\widehat{\beta_{ij}}$	$b_{00} = 0$

E. Lists

In this section, we will consider three types of lists: simple unnumbered, numbered and bulleted. The basic list is implemented with dashes "-", but it can be changed to anything with the command: `#set list(marker: [->])`.

A simple list

- bananas
- apples
- camels

coded as:

```
- bananas
- apples
- camels
```

A simple numbered list

1. bananas
2. apples
3. camels

coded as:

```
+ bananas
+ apples
+ camels
```

A plain list

bananas
apples
camels

coded as:

```
#set list(marker:[])
- bananas
- apples
- camels
```

VIII. Mathematical Typstography

Simply *beautiful*...

$$\sum_{\substack{n=12\\ n\neq i}}^{N} \lim_{\substack{uvw=12\\ n\neq ijk}} \left\langle a \middle| e^{\int g_{[i} dx} \middle| f_{j]} \right\rangle \tag{1}$$

is coded as:

```
$ sum_(n=12 \ n!=i)^N lim_(u  v  w&=12 \ n&!=i  j
k) lr(angle.l a abs(e^(integral g_(\[i)d x))f_(j\])
angle.r) $
```

and it supports functions...
Create your function: `#let ket(X) = $lr(| #X angle.r)$`, and then use it!

```
$ H ket(0) = 1/sqrt(2) ket(+) + 1/sqrt(2) ket(-) $
```

$$H|0\rangle = \frac{1}{\sqrt{2}}|+\rangle + \frac{1}{\sqrt{2}}|-\rangle \tag{2}$$

A. Display Equations

There are two ways of using expression, either inline: $\forall x \in \Gamma$ or as a display style:

$$\Pi = \sum_{k=1}^{p} |\varepsilon_k\rangle\langle\varepsilon_k| \text{ projects to } E = \text{vect}\big(\varepsilon_1, ..., \varepsilon_p\big) \quad (3)$$

is coded as follows:

```
$
Pi  =  sum_(k=1)^p  ket(epsilon_k)  bra(epsilon_k)
"projects to" E = "vect"(epsilon_1, dots, epsilon_p)
$<super_projection>
```

The difference lies in the spaces. If **both** "$" are followed with at least one space, it will be displayed, otherwise it will stay inline.
To reference this equation in the text use `@super_projection`
Please see (3)
is coded as follows:

```
Please see (@super_projection)
```

B. Equation Numbering

Consecutive Numbering: Equations within an article are numbered consecutively from the beginning of the article to the end, i.e., (1), (2), (3), (4), (5), etc.

Custom numbering: This is possible with the `numbering` function.

C. Multi-line equations and alignment

$$a = c + d \quad (4)$$
$$b = e + f \quad (5)$$

is coded as:

```
$ a &= c+d $
$ b &= e+f $
```

D. Example of a custom numbering

$$f = g \quad (7a)$$
$$f' = g' \quad (7b)$$
$$\mathcal{L}f = \mathcal{L}g \quad (7c)$$

is coded as:

```
#set math.equation(numbering: "(7a)")
#counter(math.equation).update(0)
$ f&=g $
$ f' &=g' $
$ cal(L)f &= cal(L)g $
```

E. Matrices

Matrices are created with the `mat` function. It can be used with multiple arguments, such as follows:

A simple matrix

$$\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \quad (6)$$

is coded as:

```
mat(
  0, 1 ;
  1, 0
)
```

A matrix with square brackets

$$\begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \quad (7)$$

is coded as:

```
mat(
  delim: "[",
  0 , -1 ;
  1 ,  0
)
```

A matrix with curly braces

$$\begin{Bmatrix} 1 & 0 \\ 0 & -1 \end{Bmatrix} \quad (8)$$

is coded as:

```
mat(
  delim: "{",
  1, 0 ;
  0, -1
)
```

A matrix with single verticals

$$\begin{vmatrix} a & b \\ c & d \end{vmatrix} \quad (9)$$

is coded as:

```
mat(
  delim: "|",
  a, b ;
  c, d
)
```

A matrix with double verticals

$$\begin{Vmatrix} i & 0 \\ 0 & -i \end{Vmatrix} \quad (10)$$

is coded as:

```
mat(
  delim: "||",
  i, 0 ;
  0, -i
```

```
)
```

A matrix with custom alignment

The last column is aligned differently:

$$\begin{pmatrix} a+b+c & uv & x-y & 27 \\ a+b & u+v & z & 134 \end{pmatrix} \tag{11}$$

is coded as:

```
mat(
  a + b + c, u v, x - y, 27;
  a + b, u + v, z, 134&
)
```

A matrix with vertical and horizontal rules

$$\left(\begin{array}{c|c|cc} a+b+c & uv & x-y & 27 \\ \hline a+b & u+v & z & 134 \end{array}\right) \tag{12}$$

is coded as:

```
mat(
  augment: #(
    hline: 1,
    vline: (1, 2),
  ),
  a + b + c, u v, x - y, 27 ;
  a + b, u + v, z, 134
)
```

F. Cases Structures

Cases are created using the `case` function. It is as simple as the creating a matrix:

$$z_m(t) = \begin{cases} 1, & \text{if } \beta_m(t) \\ 0, & \text{otherwise.} \end{cases}$$

is coded as follows:

```
z_m (t) = cases(
  1\, space "if" beta_m (t),
  0\, space "otherwise."
)
```

Make sure to escape characters such as “,” when calling functions as “,” are used to separate arguments. It may be possible to add “`space`” to ajust the spacing.

G. Attachs

Attachs are made easy with the widly used “`^`” and “`_`”, but Typst add a simple command to reproduce the limits behaviour:

$$d_R^{KM} = \arg\min_{d_l^{KM}} \{d_1^{KM}, ..., d_6^{KM}\}$$

is coded as follows:

```
d^(K M)_R = limits(arg min)_(d^(K M)_l){d^(K M)_1, dots, d^(K M)_6}
```

H. Text Acronyms inside equations

Simple text can be added everywhere inside equations with double quotes:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n} \left(Y_i - \widehat{Y}_i\right)^2$$

is given by:

```
"MSE" = 1/n sum_(i=1)^n (Y_i - hat(Y_i))^2
```

IX. A Final Checklist

1. Typst is cool.
2. Typst is fast.
3. Typst is Rust.
4. Make sure to reads the clear, comprehensible and nearly complete documentation at https://typst.app/docs/.
5. Enjoy typing!

References

[1] E. Example and E. Example, “Example”, *Example Example Example Example*, vol. 1, pp. 1–2, 2020.

[2] E. Example and E. Example, “Example”, *Example Example Example Example*, vol. 1, pp. 1–2, 2020.

[3] E. Example and E. Example, “Example”, *Example Example Example Example*, vol. 1, pp. 1–2, 2020.