GHENT
UNIVERSITY

Typst: a modern LaTeX replacement

Workshop for 2nd year MSc. in Photonics Student

Outline

1. Introduction
2. It’s ~~hammer~~ Typst time!
3. Let’s get started!
4. Did you say demo?
5. About your thesis

Introduction

Why?

- LaTeX is **old**

Why?

- LaTeX is **old**
- LaTeX is **hard**

Why?

- LaTeX is **old**
- LaTeX is **hard**
- LaTeX is **ugly**

Why?

- LaTeX is **old**
- LaTeX is **hard**
- LaTeX is **ugly**

```latex
\title{My first LaTeX
\author{Hubert
\date{August 2022}
\begin{document}
\maketitle
```

Why?

- LaTeX is **old**
- LaTeX is **hard**
- LaTeX is **ugly**

```latex
\title{My first LaTeX
\author{Hubert
\date{August 2022}
\begin{document}
\maketitle
```

- LaTeX is **slow**

- Typst is **modern**

Why?

- LaTeX is **old**
- LaTeX is **hard**
- LaTeX is **ugly**

LaTeX

```
\title{My first LaTeX
\author{Hubert
\date{August 2022}
\begin{document}
\maketitle
```

- LaTeX is **slow**

- Typst is **modern**
- Typst is **easy**

Why?

- LaTeX is **old**
- LaTeX is **hard**
- LaTeX is **ugly**

- Typst is **modern**
- Typst is **easy**
- Typst is **beautiful**

LaTeX

```
\title{My first LaTeX
\author{Hubert
\date{August 2022}
\begin{document}
\maketitle
```

- LaTeX is **slow**

Why?

- LaTeX is **old**
- LaTeX is **hard**
- LaTeX is **ugly**

- Typst is **modern**
- Typst is **easy**
- Typst is **beautiful**

```latex
\title{My first LaTeX
\author{Hubert
\date{August 2022}
\begin{document}
\maketitle
```

```typst
// Easy as pie
#set document(
  title: "My first Typst
  author: "Hubert Farnsworth",
)
```

- LaTeX is **slow**

Why?

- LaTeX is **old**
- LaTeX is **hard**
- LaTeX is **ugly**

- Typst is **modern**
- Typst is **easy**
- Typst is **beautiful**

LaTeX

```latex
\title{My first LaTeX
\author{Hubert
\date{August 2022}
\begin{document}
\maketitle
```

Typst

```typst
// Easy as pie
#set document(
  title: "My first Typst
  author: "Hubert Farnsworth",
)
```

- LaTeX is **slow**

- Typst is **fast**

Is LaTeX really that old?

- TeX was created in 1978 by **Donald Knuth**
  - Dude is absolutely bad-ass
  - Wrote “The Art of Computer Programming”
  - Created **METAFONT** & **Computer Modern**
  - Line-breaking algorithm
  - Document format: **DVI**
  - Created **WEB**

Is LaTeX really that old?

- TeX was created in 1978 by **Donald Knuth**
  - Dude is absolutely bad-ass
  - Wrote “The Art of Computer Programming”
  - Created **METAFONT** & **Computer Modern**
  - Line-breaking algorithm
  - Document format: **DVI**
  - Created **WEB**
- LaTeX was created in 1983 by **Leslie Lamport**
  - Set of macros of TeX
  - Easier to use
  - Closer logical structure ↔ visual structure

Is it really that hard?

- LaTeX is known for it’s cryptic errors

```
! Undefined control sequence.
\enit@setresumekeys ...it@toks }\ifnum \enit@type
    =\z@ #3\def \enit@noexcs {...
```

Is it really that hard?

- LaTeX is known for it's cryptic errors

```
! Undefined control sequence.
\enit@setresumekeys ...it@toks }\ifnum \enit@type
    =\z@ #3\def \enit@noexcs {...
```

- People just Google for info

Is it really that hard?

- LaTeX is known for it's cryptic errors

```
! Undefined control sequence.
\enit@setresumekeys ...it@toks }\ifnum \enit@type
    =\z@ #3\def \enit@noexcs {...
```

- People just Google for info
- The documentation is often hard to search

LaTeX is ugly?

> “Because it [LaTeX] is so hacky and messy. […]”
>
> — u/atloomis

- Have you ever wondered what goes on in your `documentclass`?

```latex
\def\@citex[#1]#2{%
\let\@citea\@empty
\@cite{\@for\@citeb:=#2\do
  {\@citea\def\@citea{], []}%
  \edef\@citeb{\expandafter\@firstofone\@citeb\@empty}%
```

It's ~~hammer~~ Typst time!

What sets Typst apart?

- A **real** programming language

Typst

```typst
#let fib(n) = {
    if n <= 1 {
        1
    } else {
        fib(n - 1) + fib(n - 2)
    }
}
```

What sets Typst apart?

- A **real** programming language
- With **powerful** markup syntax

```typst
#let fib(n) = {
  if n <= 1 {
    1
  } else {
    fib(n - 1) + fib(n - 2)
  }
}
```

Typst

```typst
==== My paragraph
Hello, world!
*This text is in bold*.
_And this one is emphasized_.
#strike[This one is
  struck through].
```

Typst

GHENT UNIVERSITY

What sets Typst apart?

- A simple **free** web-app for **everything**
  - Collaborative editing
  - Instant preview
  - Cloud storage

What sets Typst apart?

- A simple **free** web-app for **everything**
  - Collaborative editing
  - Instant preview
  - Cloud storage
- Or running on your PC using a **single** binary
  - macOS, Linux, or Windows
  - No dependencies
  - Easy to install
  - Easy to update

What sets Typst apart?

- A simple **free** web-app for **everything**
  - Collaborative editing
  - Instant preview
  - Cloud storage
- Or running on your PC using a **single** binary
  - macOS, Linux, or Windows
  - No dependencies
  - Easy to install
  - Easy to update
- **Incredible** documentation

What sets Typst apart?

- A simple **free** web-app for **everything**
  - Collaborative editing
  - Instant preview
  - Cloud storage
- Or running on your PC using a **single** binary
  - macOS, Linux, or Windows
  - No dependencies
  - Easy to install
  - Easy to update
- **Incredible** documentation
- **Packages** at your fingertips

Let's get started!

The basics

- A Typst document is a **markup** with **code**

The basics

- A Typst document is a **markup** with **code**
- The markup is the default mode or when surrounded in `[` and `]`.

The basics

- A Typst document is a **markup** with **code**
- The markup is the default mode or when surrounded in `[` and `]`.
- The `#` character is used to switch to code mode.

The basics

- A Typst document is a **markup** with **code**
- The markup is the default mode or when surrounded in `[` and `]`.
- The `#` character is used to switch to code mode.

```
This is markup
#this-is-code()
#[ This is also markup ]
```

The basics (cont.)

- Heading levels are defined with `=` characters

Typst

```
= Heading 1
== Heading 2
=== Heading 3
```

The basics (cont.)

- Heading levels are defined with `=` characters

```typst
= Heading 1
== Heading 2
=== Heading 3
```

- Lists are defined with `-` characters

```typst
- Item 1
- Item 2
  - Item 2.1
```

The basics (cont.)

- Numbered lists are defined with `+` characters

```typst
+ Item 1
+ Item 2
  + Item 2.1
```

The basics (cont.)

- Numbered lists are defined with `+` characters

```typst
+ Item 1
+ Item 2
  + Item 2.1
```

- Strong emphasis is done with `*` and `_` characters

```typst
*This is strong*
_This is emphasized_
```

The basics (cont.)

- You can create labels using the `<label>` syntax:

Typst

```typst
= Chapter 1 <my-label>
```

The basics (cont.)

- You can create labels using the `<label>` syntax:

```typst
= Chapter 1 <my-label>
```

- And you can reference them using the `@` syntax:

```typst
@my-label // This creates a clickable link
```

The basics (cont.)

- You can create labels using the `<label>` syntax:

```typst
= Chapter 1 <my-label>
```

- And you can reference them using the `@` syntax:

```typst
@my-label // This creates a clickable link
```

- It works the same for bibliographies!

```typst
@my-bib-entry // This creates a clickable link
```

The basics (cont.)

- You can insert comments using `//` and `/* */`

```typst
// This is a comment
/* This is a block comment */
```

The basics (cont.)

- You can insert comments using `//` and `/* */`

Typst

```
// This is a comment
/* This is a block comment */
```

- You can insert code using the `#` character

Typst

```
#let x = 1
#let y = 2
#let z = x + y
```

The basics (cont.)

- You can insert a block of code using the `#` character

Typst

```typst
#{
   let x = 1
   let y = 2
}
```

The basics (cont.)

- You can insert a block of code using the `#` character

```typst
#{
    let x = 1
    let y = 2
}
```

- You can declare functions and variables using the `let` keyword

```typst
#let add(a, b) = a + b
#let c = add(5, 6)
```

Bibliographies

- Please use **Zotero** for managing your bibliographies

Bibliographies

- Please use **Zotero** for managing your bibliographies
- You can export your bibliography to a `.bib` file

Bibliographies

- Please use **Zotero** for managing your bibliographies
- You can export your bibliography to a `.bib` file
- Or you can use **Hayagriva**'s `.yaml` files

Bibliographies

- Please use **Zotero** for managing your bibliographies
- You can export your bibliography to a `.bib` file
- Or you can use **Hayagriva**’s `.yaml` files
- You can then import them in your Typst document

Typst

```typst
#bibliography("my-bibliography.bib", style: "ieee")
```

Bibliographies

- Please use **Zotero** for managing your bibliographies
- You can export your bibliography to a `.bib` file
- Or you can use **Hayagriva**'s `.yaml` files
- You can then import them in your Typst document

```typst
#bibliography("my-bibliography.bib", style: "ieee")
```

- You can then cite them using the `@` syntax

```typst
@my-bib-entry // Will display like [1] and be clickable.
```

Bibliographies

- Please use **Zotero** for managing your bibliographies
- You can export your bibliography to a `.bib` file
- Or you can use **Hayagriva**’s `.yaml` files
- You can then import them in your Typst document

```typst
#bibliography("my-bibliography.bib", style: "ieee")
```

- You can then cite them using the `@` syntax

```typst
@my-bib-entry // Will display like [1] and be clickable.
```

- Everything will be formatted automatically!

Images and figures

- You can insert images using the `#image` function

```typst
#image("my-image.png")
```

Images and figures

- You can insert images using the `#image` function

```typst
#image("my-image.png")
```

Typst

- And wrap them in a figure using the `#figure` function

```typst
#figure(caption: "My caption")[
  #image("my-image.png")
] <my-image>
```

Typst

Images and figures

- You can insert images using the `#image` function

```typst
#image("my-image.png")
```

- And wrap them in a figure using the `#figure` function

```typst
#figure(caption: "My caption")[
  #image("my-image.png")
] <my-image>
```

- It automatically detects the type of figure (image, table, etc.)

Images and figures

- You can insert images using the `#image` function

```typst
#image("my-image.png")
```

- And wrap them in a figure using the `#figure` function

```typst
#figure(caption: "My caption")[
  #image("my-image.png")
] <my-image>
```

- It automatically detects the type of figure (image, table, etc.)
- You can reference it by giving it a label and referecing it using the `@` syntax.

Equations

- Two types: inline and display
- Inline equations are surrounded by `$` characters: $x^2 + y^2 = z^2$

Typst

```typst
$x^2 + y^2 = z^2$
```

Equations

- Two types: inline and display
- Inline equations are surrounded by `$` characters: $x^2 + y^2 = z^2$

```typst
$x^2 + y^2 = z^2$
```

- Display equations are surrounded by `$` characters with a space:

$$x^2 + y^2 = z^2$$

```typst
$ x^2 + y^2 = z^2 $
```

Equations

- Two types: inline and display
- Inline equations are surrounded by `$` characters: $x^2 + y^2 = z^2$

```typst
$x^2 + y^2 = z^2$
```

- Display equations are surrounded by `$` characters with a space:

$$x^2 + y^2 = z^2$$

```typst
$ x^2 + y^2 = z^2 $
```

- You can also give them a label and reference them.

Equations

- Two types: inline and display
- Inline equations are surrounded by `$` characters: $x^2 + y^2 = z^2$

```typst
$x^2 + y^2 = z^2$
```

- Display equations are surrounded by `$` characters with a space:

$$x^2 + y^2 = z^2$$

```typst
$ x^2 + y^2 = z^2 $
```

- You can also give them a label and reference them.
- Syntax is similar to LaTeX but with a few differences.

Outlines & queries

- Outlines are built from a query

Outlines & queries

- Outlines are built from a query
- Queries allow you to ask questions of your document

```typst
#locate(loc => {
  query(heading.where(level: 1), loc)
})
```

Outlines & queries

- Outlines are built from a query
- Queries allow you to ask questions of your document

```typst
#locate(loc => {
  query(heading.where(level: 1), loc)
})
```

- And are made easy to use

```typst
#outline(target: figure.where(kind: image))
```

Packages

- Packages are a way to extend Typst

Packages

- Packages are a way to extend Typst
- They are written in Typst itself

Packages

- Packages are a way to extend Typst
- They are written in Typst itself
- They are easy to import

```typst
#import "@preview/polylux:0.3.1": *
```

Packages

- Packages are a way to extend Typst
- They are written in Typst itself
- They are easy to import

```typst
#import "@preview/polylux:0.3.1": *
```

- And easy to use

```typst
#show: codly-init.with()
#codly()
```

Packages

- Packages are a way to extend Typst
- They are written in Typst itself
- They are easy to import

```typst
#import "@preview/polylux:0.3.1": *
```

- And easy to use

```typst
#show: codly-init.with()
#codly()
```

- Downloaded on demand

show rules, and set commands

- show rules are used to change how an object is displayed

```typst
#show link: it => text(color: blue, it)
#link("https://www.google.com/") will be blue.
```

show rules, and set commands

- show rules are used to change how an object is displayed

```typst
#show link: it => text(color: blue, it)
#link("https://www.google.com/") will be blue.
```

- set commands are used to change the state of a function

```typst
#set text(font: "New Computer Modern")
This text will be in New Computer Modern.
```

`show` rules, and `set` commands

- `show` rules are used to change how an object is displayed

```typst
#show link: it => text(color: blue, it)
#link("https://www.google.com/") will be blue.
```

- `set` commands are used to change the state of a function

```typst
#set text(font: "New Computer Modern")
This text will be in New Computer Modern.
```

- Rules are applied in order of appearance

`show` rules, and `set` commands

- `show` rules are used to change how an object is displayed

```typst
#show link: it => text(color: blue, it)
#link("https://www.google.com/") will be blue.
```

- `set` commands are used to change the state of a function

```typst
#set text(font: "New Computer Modern")
This text will be in New Computer Modern.
```

- Rules are applied in order of appearance
- Rules are scoped to the current block

Nifty packages

- `@preview/polylux` for making slides (like this one)
- `@preview/codly` for beautiful code blocks
- `@preview/cetz` for creating diagrams
- `@preview/tablex` for creating beautiful tables
- `@preview/glossarium` for creating glossaries
- `@preview/lemmify` and `@preview/ctheorems` for creating theorems
- `@preview/jogs` to run JS code
- `@preview/pyrunner` to run Python code from Typst
- So many more

Did you say demo?

About your thesis

GHENT UNIVERSITY

Use these tools

- Zotero for managing your bibliography
- Typst for writing your thesis
- https://draw.io for creating diagrams
- The Typst Discord server: https://discord.gg/2uDybryKPe

Thanks for coming!

- https://typst.app
- https://discord.gg/2uDybryKPe
- https://github.com/typst/typst
- https://github.com/Dherse/masterproef
- https://github.com/Dherse/ugent-templates
- Questions?

Sébastien d'Herbais de Thun

Student

E sebastien.dherbaisdethun@ugent.be

Universiteit Gent

@ugent

@ugent

Ghent University

Sébastien d'Herbais de Thun

www.ugent.be