

Typst – Hat L^AT_EX ausgedient?



Eine kurze Einführung in Typst

Tristan Pieper tristan.pieper@uni-rostock.de · 2. Juni 2023

Inhaltsverzeichnis

1. Kurzes Kennenlernen	3
2. Probleme von L ^A T _E X.....	4
3. Die Lösung aller Probleme(?)	13
4. Die Web-App	17
5. Grundlegende Formatierung	19
6. Die Typst-Dokumentation...	45
7. Eigene Templates und Skripts	61
8. Typst kann noch mehr!	67
9. Abschluss und Weiteres	71

1. Kurzes Kennenlernen

2. Probleme von L^AT_EX

2.1. Alles begann mit...



Donald E. Knuth¹ (geb. 10. Januar 1938)

2.2. Dann kam...



Leslie Lamport¹ (geb. 7. Februar 1941)

2.3. Die Probleme

1. Riesige Programmgröße
2. Auswahl an Compilern
3. Unverständliche Fehler

2.4. Größe des Programms

```
% du -sch /usr/share/texmf-dist/* | sort -hr
2,5G    insgesamt
1,9G    /usr/share/texmf-dist/fonts
499M    /usr/share/texmf-dist/tex
58M     /usr/share/texmf-dist/scripts
44M     /usr/share/texmf-dist/tex4ht
24M     /usr/share/texmf-dist/bibtex
15M     /usr/share/texmf-dist/metapost
7,5M    /usr/share/texmf-dist/dvips
3,8M    /usr/share/texmf-dist/xindy
3,4M    /usr/share/texmf-dist/ls-R
2,6M    /usr/share/texmf-dist/asymptote
1,7M    /usr/share/texmf-dist/context
516K    /usr/share/texmf-dist/omega
344K    /usr/share/texmf-dist/makeindex
```

Verglichen mit 21MB des Typst-Compilers...

```
% du -sch /usr/bin/typst
21M    /usr/bin/typst
21M    insgesamt
```

2.5. Die Vielfalt

„ \LaTeX “ ist kein Programm, sondern:

- pdfTeX
- LuaTeX
- XeTeX
- MikTeX
- KaTeX
- ...

2.6. Beispiel- Fehlernmeldung (Typst)

Typst:

\$a+b

```
error: expected dollar sign
└ test.typ:1:5
  1 | $a+b
    ^
```

2.7. Beispiel-Fehlermeldung (L^AT_EX)

L^AT_EX:

```
\documentclass{article}
```

```
\begin{document}
```

\$a+b

```
\end{document}
```

```
Latexmk: This is Latexmk, John Collins, 17 Mar. 2022. Version 4.77,
version: 4.77.
Latexmk: applying rule 'pdflatex'...
Rule 'pdflatex': File changes, etc:
    Changed files, or newly in use since previous run(s):
    /path/Desktop/Projekte/Typst/typst-seminar/.lt/test.tex
    test.tex
Rule 'pdflatex': The following rules & subrules became out-of-date:
    pdflatex
-----
Run number 1 of rule 'pdflatex'
-----
-----
Running 'pdflatex -synctex=1 -interaction=nonstopmode -file-line-
error -recorder "/path/Desktop/Projekte/Typst/typst-seminar/.lt/
test.tex"'
-----
This is pdfTeX, Version 3.141592653-2.6-1.40.24 (TeX Live 2022/Arch
Linux) (preloaded format=pdflatex)
  restricted \write18 enabled.
entering extended mode
(/path/Desktop/Projekte/Typst/typst-seminar/.lt/test.tex
LaTeX2e <2021-11-15> patch level 1
L3 programming layer <2022-04-10> (/usr/share/texmf-dist/tex/latex/
base/article.cls
Document Class: article 2021/10/04 v1.4n Standard LaTeX document
class
(/usr/share/texmf-dist/tex/latex/base/size10.clo)) (/usr/share/
texmf-dist/tex/latex/l3backend/l3backend-pdftex.def) (./test.aux)
/path/Desktop/Projekte/Typst/typst-seminar/.lt/test.tex:5: Missing
$ inserted.
<inserted text>
                $
l.5

[1{/var/lib/texmf/fonts/map/pdftex/updmap/pdftex.map}] (./
test.aux) )
(see the transcript file for additional information)</usr/share/
texmf-dist/fonts/type1/public/amsfonts/cm/cmr10.pfb>
Output written on test.pdf (1 page, 13646 bytes).
SyncTeX written on test.synctex.gz.
Transcript written on test.log.
Latexmk: If appropriate, the -f option can be used to get latexmk
  to try to force complete processing.
Latexmk: Getting log file 'test.log'
Latexmk: Examining 'test.fls'
Latexmk: Examining 'test.log'
Latexmk: Log file says output to 'test.pdf'
Latexmk: Errors, so I did not complete making targets
Collected error summary (may duplicate other messages):
```

2.8. Mehr Beispiel-Fehlermeldung

Typst	LATEX
<pre>#set par(leading: [Hello]) ^^^^^^ expected integer, found content</pre>	\baselineskip=Hello Missing number, treated as zero. Illegal unit of measure (pt inserted).
<pre>#heading() ^ missing argument: body</pre>	\section Missing \endcsname inserted. Missing \endcsname inserted. ...

(aus: Mädje, Laurenz: „Typst – A Programmable Markup Language for Typesetting.“ Masterarbeit an der TU Berlin, 2022.)

3. Die Lösung aller Probleme(?)

3.1. Typst



Martin Haug
(Webentwicklung)

Laurenz Mädje
(Compilerentwicklung)

<https://typst.app/about/>, (letzter Zugriff: 26.05.2023, 10:16)

- 2019 Projektstart an TU Berlin
- entstanden aus Frustration über LaTeX

3.2. Das Ziel

„Während bestehende Lösungen langsam, schwer zu bedienen oder einschränkend sind, ist Typst sorgfältig entworfen, um leicht erlernbar, flexibel und schnell zu sein. Dafür haben wir eine komplett eigene Markupsprache und Textsatzengine von Grund auf entwickelt. Dadurch sind wir in allen Bereichen des Schreib- und Textsatzprozesses innovationsfähig.“

<https://www.tu.berlin/entrepreneurship/startup-support/unsere-startups/container-profile/startups-2023-typst>, (letzter Zugriff: 03.05.2023, 10:13)

3.3. Ein kleiner Vergleich

LaTeX	Typst	Ergebnis
<code>\documentclass{article}</code>	+ Dies	1. Dies
<code>\begin{document}</code>	+ Ist	2. Ist
<code>\begin{enumerate}</code>	+ Eine	3. Eine
<code> \item Dies</code>	+ Liste!	4. Liste!
<code> \item Ist</code>		
<code> \item Eine</code>		
<code> \item Liste!</code>		
<code>\end{enumerate}</code>		
<code>\end{document}</code>		

4. Die Web-App

4.1. Ab ans Werk!

Vorteile:

- alle Dateien online
- verschiedene Projekte erstellbar
- guter online Editor
- als Team gleichzeitig an Dateien arbeiten
- eingebaute Dokumentation

<https://typst.app/>

Temporäre Accounts (mit $1 \leq N \leq 15$):

- E-Mail: typstseminarN@byom.de
- Passwort: typstseminarN

Link zur Präsentation: <https://github.com/survari/typst-seminar>

5. Grundlegende Formatierung

5.1. Überschriften

```
// Hier ein Kommentar,  
// er wird ignoriert!
```

= Überschrift 1!
== Überschrift 2!
== Überschrift 3!
Text

Neuer Abstand!

1. Überschrift 1!

1.1. Überschrift 2!

1.1.1. Überschrift 3!

Text

Neuer Absatz!

5.2. Absätze

= Mein cooler Titel

In Typst beginnt ein neuer Absatz, sobald im Code eine freie Zeile steht.

Leider sind standardmäßig Absätze linksbündig und nicht Blocksatz. Wie man das ändert, lernen wir gleich!

1. Mein cooler Titel

In Typst beginnt ein neuer Absatz, sobald im Code eine freie Zeile steht.

Leider sind standardmäßig Absätze linksbündig und nicht Blocksatz. Wie man das ändert, lernen wir gleich!

5.3. Listen

- + Eine nummerierte Liste
 - + Kann sehr schön sein!
 - 1. So geht sie auch!
 - 2. Jawohl!
 - Und hier nicht-nummeriert!
 - Ganz ohne Nummern!
- 1. Eine nummerierte Liste
 - 2. Kann sehr schön sein!
 - 1. So geht sie auch!
 - 2. Jawohl!
 - Und hier nicht-nummeriert!
 - Ganz ohne Nummern!

5.4. Schriftart

```
#text(font: "Arial", [Hallo Welt!])
```

Hallo Welt!

```
#text(font: "Courier New", [Hallo  
Welt!])
```

Hallo Welt!

```
#text(font: "New Computer Modern",  
[Hallo Welt!])
```

Hallo Welt!

5.5. Content, Strings, Scripts

- zwei Arten von Inhalten in Typst:
 - Content in [...]
 - Scripts in {...}
 - von Content zu Script mit #
- jedes Dokument ist grundlegend Content

```
#strong([Hier ist Content  
in Script.])
```

Hier ist Content.

```
#{  
    strong([Fett!])  
    [Auch hier ist Content  
in Script!]  
}
```

Fett!Auch hier ist Content in
Script!

0.75

```
#{ 3/4 }
```

5.6. Text¹

```
*Hello!* #strong([Hello!])  
  
_Hello!_ #emph([Hello!])  
  
Hello!#super([Hello!])  
  
Hello!#sub([Hello!])  
  
#text(fill: red, [Hello rot!])  
  
#text(fill: rgb("#ff00ff"), [Hello  
pink!])  
  
#text(fill: rgb("#ff00ff"), strong([Hello  
pink!]))
```

Hello! Hello!

Hello! Hello!

Hello!^{Hello!}

Hello!_{Hello!}

Hallo rot!

Hallo pink!

Hallo pink!

5.7. Ausrichtung

```
#align(left, [Hallo!])          Hallo!
#align(center, [Hallo!])        Hallo!
#align(right, [Hallo!])         Hallo!
#align(right, strong([Hallo!])) Hallo!
```

Aufgabe 1

Wie realisiert man das Folgende in Typst?

Ein Brief an Lorem

 Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do
 eiusmod tempor incididunt ut labore et dolore magna aliquam
 quaerat.

Lösung 1

```
#align(center, emph[Ein Brief an Lorem])  
#lorem(20)
```

5.8. Abstände #1

```
#align(right, [Vertikaler])  
#v(2cm)  
Horizontaler #h(2cm) Abstand
```

Vertikaler

Horizontaler Abstand

5.9. Abstände (Vertikal)

Listen-Beispiel von Folie 21 mit vertikalen Abständen.

- + Eine nummerierte Liste
 - + Kann sehr schön sein!
- #v(2em)
1. So geht sie auch!
 2. Jawohl!
- #v(2em)
- Und hier nicht-nummeriert!
 - Ganz ohne Nummern!

Listen-Beispiel von Folie 21 mit vertikalen Abständen.

1. Eine nummerierte Liste
 2. Kann sehr schön sein!
-
1. So geht sie auch!
 2. Jawohl!
-
- Und hier nicht-nummeriert!
 - Ganz ohne Nummern!

5.10. Abstände (Horizontal)

- + Gott kommen aufgrund seines vollkommenen Wesens alle vollkommenen Eigenschaften zu.
- + Zu existieren ist vollommener als nicht zu existieren.
- + Also ist Existenz eine vollkommene Eigenschaft.
- + Also existiert Gott. #h(1fr) QED

1. Gott kommen aufgrund seines vollkommenen Wesens alle vollkommenen Eigenschaften zu.
2. Zu existieren ist vollommener als nicht zu existieren.
3. Also ist Existenz eine vollkommene Eigenschaft.
4. Also existiert Gott.

QED

5.11. Bilder

```
#image(height: 50%, "leslie_lamport.png")
```



5.12. `#block()` und `#box()`

Das lässt sich mit `#block` machen: `#block(stroke: black, inset: 0.5em, ['#block()')` erzeugt einen Zeilenumbruch und lässt sich über Seiten brechen. Es hat viele optionale Argumente.]
So sieht's aus.

Das lässt sich mit `#block` machen:

`#block()` erzeugt einen Zeilenumbruch und lässt sich über Seiten brechen. Es hat viele optionale Argumente.

So sieht's aus.

5.13. `#block()` und `#box()`

Hingegen: `#box(stroke: black, inset: 2pt, [`#box()`])` erzeugt `#box(stroke: (bottom: black), inset: 2pt, [keinen])` Zeilenumbruch und lässt sich `#box(stroke: (bottom: black), inset: 2pt, [nicht])` über Seiten brechen. Man kann damit aber Dinge zum Beispiel umrahmen. Zum Unterstreichen sollte man aber eher `#underline[`#underline`]` benutzen.

Hingegen: `#box()` erzeugt keinen Zeilenumbruch und lässt sich nicht über Seiten brechen. Man kann damit aber Dinge zum Beispiel umrahmen. Zum Unterstreichen sollte man aber eher `#underline` benutzen.

5.14. Tabellen

```
#table(  
  columns: (auto, 3cm, auto),  
  [Hallo1!],  
  [2a],  
  [Hallo3!],  
  [Welt1!],  
  [2b],  
  [Welt3!]  
)
```

Hallo1!	2a	Hallo3!
Welt1!	2b	Welt3!

5.15. Mathematik

```
$ sum_(k=0)^n k = 1 + ... + n $  
  
$ A = pi r^2 $  
  
$ "area" = pi dot.op "radius"^2 $  
  
$ cal(A) :=  
  { x in RR | x "is natural" } $  
  
$ frac(a^2, 2) $
```

$$\sum_{k=0}^n k = 1 + \dots + n$$

$$A = \pi r^2$$

$$\text{area} = \pi \cdot \text{radius}^2$$

$$\mathcal{A} := \{x \in \mathbb{R} \mid x \text{ is natural}\}$$

$$\frac{a^2}{2}$$

Aufgabe 2

Wie realisiert man das Folgende in Typst?

Formel	Zugeschrieben
$a^2 + b^2 = c^2$	Pythagoras
$c \leq a + b$	Unbekannt

Lösung 2

```
#table(columns: (auto, auto),  
       strong[Formel], strong[Zugeschrieben],  
       $a^2 + b^2 = c^2$, [Pythagoras],  
       $c <= a + b$, [Unbekannt])
```

5.16. Set-Regeln¹

Hier ist noch die Standard-Schriftart! Q

```
#set text(font: "New Computer Modern", fill: blue)
```

Q Ab jetzt ist alles vollkommen in der anderen Schriftart und sogar blau!

```
#set par(first-line-indent: 1.5em, justify: true)
```

Ab jetzt wird jede erste Zeile eines Absatzes eingerückt und Blocksatz!

Wirklich, versprochen!
`#lorem(20)`

Hier ist noch die Standard-Schriftart! Q

Q Ab jetzt ist alles vollkommen in der anderen Schriftart und sogar blau!

Ab jetzt wird jede erste Zeile eines Absatzes eingerückt und Blocksatz!

Wirklich, versprochen! Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliquam quaerat.

5.17. Show-Regeln¹

```
#show heading: set text(red)
```

==== Hallo!

===== Welt!

```
// aus dem offiziellem  
// Typst-Tutorial  
#show "Project": smallcaps  
#show "badly": "great"
```

We started Project in 2019
and are still working on it.
Project is progressing badly.

5.17.1. Hallo!

5.17.1.1. Welt!

We started PROJECT in
2019 and are still
working on it. PROJECT
is progressing great.

Aufgabe 3

Wie realisiert man das Folgende in Typst? Hinweis: Die show-Regel ist nicht auf `#lorem()` anwendbar.

In **Typst** sind show- und set-Regeln sehr mächtig. Kann man Sie, kann man auch **Typst**. **Typst** soll immer fett gedruckt werden. Der Absatz ist Blocksatz. *Kleine Wiederholung: Wie macht man in Typst nochmal etwas kursiv?*

Lösung 3

```
#set par(justify: true)
#show "Typst": strong
```

In Typst sind show- und set-Regeln sehr mächtig. Kann man Sie, kann man auch Typst. Typst soll immer fett gedruckt werden. Der Absatz ist Blocksatz. *Kleine Wiederholung: Wie macht man in Typst nochmal etwas kursiv?*

```

\documentclass[14pt,a4paper]{extarticle}
\usepackage{bold-extra}
\usepackage{amssymb}
\usepackage[T1]{fontenc}
\usepackage[paperwidth=22cm,paperheight=5.5cm,
    left=0.5cm,right=0.5cm,top=0.5cm,
    bottom=0.5cm]{geometry}

\pagenumbering{gobble}
\setlength{\parskip}{0.65em}
\setlength{\parindent}{0pt}

\begin{document}
    \noindent\textbf{\textsf{Definition 1.}}\\
\textit{Sei  $D \subsetneq \mathbb{R}$  und sei  $f: D \rightarrow \mathbb{R}$  eine Funktion.  $f$  ist stetig in  $x_0 \in D$  genau dann, wenn die folgende Aussage gilt:}

    \textit{Für alle  $\epsilon > 0$  existiert ein  $\delta > 0$ , sodass  $|f(x) - f(x_0)| < \epsilon$  für alle  $x \in D$  mit  $|x - x_0| < \delta$ .}

    \textit{Oder Alternativ:  $\forall \epsilon > 0 \exists \delta > 0 \forall x \in D : |y - y_0| < \delta \Rightarrow |f(x) - f(x_0)| < \epsilon$ }

    \bigskip
    (\LaTeX)
\end{document}

```

#set page(margin: 0.5cm, width: 22cm, height: 5.5cm)
#set text(size: 14pt, font: "New Computer Modern")
#set par(justify: true)

#smallcaps([Definition 1.]) _Sei $D \subsetneq \mathbb{R}$
und sei $f: D \rightarrow \mathbb{R}$ eine Funktion. f ist stetig in
 $x_0 \in D$ genau dann, wenn die folgende Aussage
gilt:_

_Für alle $\epsilon > 0$ existiert ein $\delta > 0$,
sodass $|f(x) - f(x_0)| < \epsilon$ für alle $x \in D$
mit $|x - x_0| < \delta$._

_Oder Alternativ: $\forall \epsilon > 0 \exists \delta > 0 \forall x \in D : |y - y_0| < \delta \Rightarrow |f(x) - f(x_0)| < \epsilon$ _

#v(lem)
(Typst)

DEFINITION 1. Sei $D \subseteq \mathbb{R}$ und sei $f : D \rightarrow \mathbb{R}$ eine Funktion. f ist stetig in $x_0 \in D$ genau dann, wenn die folgende Aussage gilt:

Für alle $\epsilon > 0$ existiert ein $\delta > 0$, sodass $|f(x) - f(x_0)| < \epsilon$ für alle $x \in D$ mit $|x - x_0| < \delta$.

Oder Alternativ: $\forall \epsilon > 0 \exists \delta > 0 \forall x \in D : |y - y_0| < \delta \Rightarrow |f(x) - f(x_0)| < \epsilon$

(LATEX)

DEFINITION 1. Sei $D \subseteq \mathbb{R}$ und sei $f : D \rightarrow \mathbb{R}$ eine Funktion. f ist stetig in $x_0 \in D$ genau dann, wenn die folgende Aussage gilt:

Für alle $\varepsilon > 0$ existiert ein $\delta > 0$, sodass $|f(x) - f(x_0)| < \varepsilon$ für alle $x \in D$ mit $|x - x_0| < \delta$.

Oder Alternativ: $\forall \varepsilon > 0 \exists \delta > 0 \forall x \in D : |y - y_0| < \delta \Rightarrow |f(x) - f(x_0)| < \varepsilon$

(Typst)

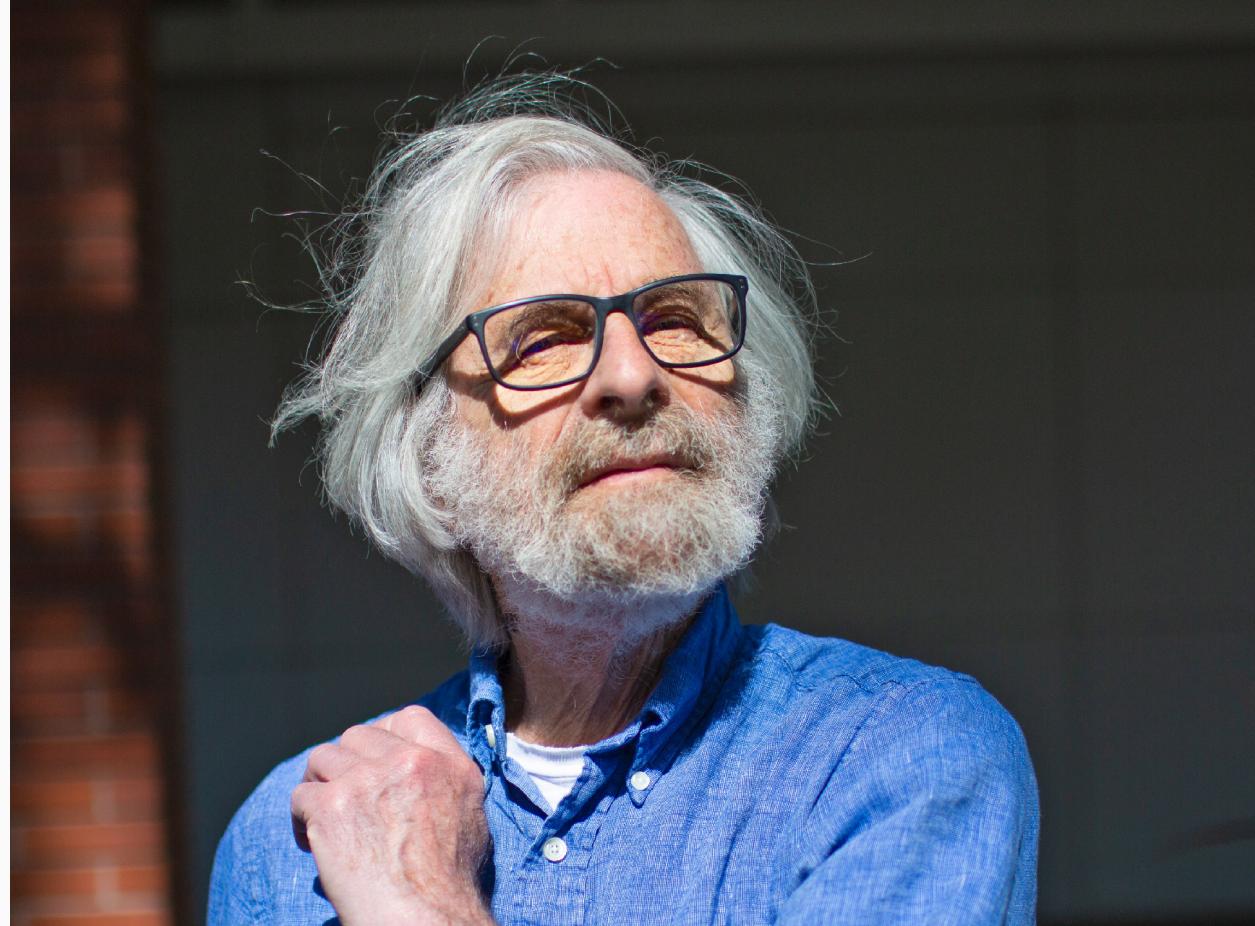
6. Die Typst-Dokumentation...

- <https://typst.app/docs> als Nachschlagwerk
- Dokumentation ist wichtig!

6.1. Dokumentations-Beispiel: image-Funktion

image kann noch ein bisschen was!

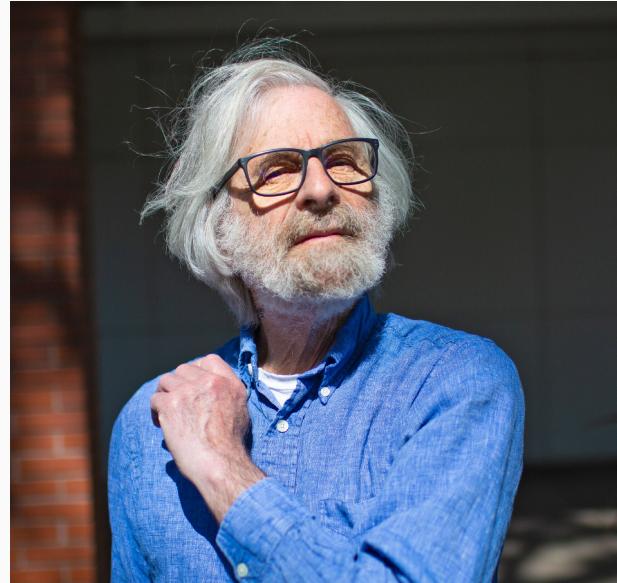
```
#image("leslie_lamport.png")
```



6.2. Dokumentations-Beispiel: image-Funktion

image kann noch ein bisschen was!

```
#image(height: 50%, "leslie_lamport.png")
```



6.3. Dokumentation Beispiel: `image`- Funktion

`image` kann noch ein bisschen was!

```
#image(fit: "stretch", width: 100%, height: 100%,  
"leslie_lamport.png")
```



[Overview](#)[Tutorial](#)[Reference](#)[Changelog](#)[Community](#)

Overview

Welcome to Typst's documentation! Typst is a new markup-based typesetting system for the sciences. It is designed to be an alternative both to advanced tools like LaTeX and simpler tools like Word and Google Docs. Our goal with Typst is to build a typesetting tool that is highly capable *and* a pleasure to use.

This documentation is split into two parts: A beginner-friendly tutorial that introduces Typst through a practical use case and a comprehensive reference that explains all of Typst's concepts and features.

We also invite you to join the community we're building around Typst. Typst is still a very young project, so your feedback is more than valuable.



Tutorial

Step-by-step guide to help you get started.



Reference

Details about all syntax, concepts, types, and functions.

image

☰ > Overview

Results

Image FUNCTION

Formatting CHAPTER

Box FUNCTION

Baseline PARAMETER OF
BOX

Padding FUNCTION

Background PARAMETER
OF PAGE

Table FUNCTION

Path PARAMETER OF IMAGE

Width PARAMETER OF
IMAGEHeight PARAMETER OF
IMAGE[Overview](#)[Tutorial](#)[Reference](#)[Changelog](#)[Community](#)

Overview

Welcome to Typst's documentation! Typst is a new markup-based typesetting system for the sciences. It is designed to be an alternative both to advanced tools like LaTeX and simpler tools like Word and Google Docs. Our goal with Typst is to build a typesetting tool that is highly capable *and* a pleasure to use.

This documentation is split into two parts: A beginner-friendly tutorial that introduces Typst through a practical use case and a comprehensive reference that explains all of Typst's concepts and features.

We also invite you to join the community we're building around Typst. Typst is still a very young project, so your feedback is more than valuable.



Tutorial

Step-by-step guide to help you get started.



Reference

Details about all syntax, concepts, types, and functions.

[Overview](#)[Tutorial](#)[Reference](#)[LANGUAGE](#)[Syntax](#)[Styling](#)[Scripting](#)[Types](#)[CONTENT](#)[Text](#)[Math](#)[Layout](#)[Visualize](#)[Circle](#)[Ellipse](#)[Image](#)[Line](#)[Path](#)[Polygon](#)[Rectangle](#)[Square](#)

image Element

A raster or vector graphic.

Supported formats are PNG, JPEG, GIF and SVG.

Example

```
#figure(  
  image("molecular.jpg", width: 80%),  
  caption: [  
    A step in the molecular testing  
    pipeline of our lab.  
  ],  
)
```



Figure 1: A step in the molecular testing pipeline of our lab.

Parameters ?

```
image(  
  string,  
  width: auto relative length,
```

[ON THIS PAGE](#)[Summary](#)[Parameters](#)[path](#)[width](#)[height](#)[alt](#)[fit](#)

Parameters ⓘ

```
image(  
    string ,  
    width: auto | relative length ,  
    height: auto | relative length ,  
    alt: none | string ,  
    fit: string ,  
) -> content
```

path string Required Positional ⓘ

width `auto` or `relative length` *Settable* ?

The width of the image.

height `auto` or `relative length` *Settable* ?

The height of the image.

alt `none` or `string` *Settable* ?

A text describing the image.

fit `string` *Settable* ?

How the image should adjust itself to a given area.

- `"cover"` The image should completely cover the area. This is the default.
- `"contain"` The image should be fully contained in the area.
- `"stretch"` The image should be stretched so that it exactly fills the area, even if this means that the image will be distorted.

Beispiel bei #enum():

numbering string or function *Settable* 

How to number the enumeration. Accepts a [numbering pattern or function](#).

If the numbering pattern contains multiple counting symbols, they apply to nested enums. If given a function, the function receives one argument if `full` is `false` and multiple arguments if `full` is `true`.

› View example

Beispiel bei #enum():

numbering string or function Settable 

How to number the enumeration. Accepts a [numbering pattern or function](#).

If the numbering pattern contains multiple counting symbols, they apply to nested enums. If given a function, the function receives one argument if `full` is `false` and multiple arguments if `full` is `true`.

▼ View example

```
#set enum(numbering: "1.a")  
+ Different  
+ Numbering  
  + Nested  
  + Items  
+ Style  
  
#set enum(numbering: n => super[#n])  
+ Superscript  
+ Numbering!
```

- 1) Different
- 2) Numbering
 - a) Nested
 - b) Items
- 3) Style

- ¹ Superscript
- ² Numbering!

Aufgabe 4

Welcher Parameter kann durch eine set-Regel wie verändert werden, damit man eine nummerierte Listen (#enum()) wie folgt formatieren kann? Die Dokumentation hilft!

- I. Element 1
- II. Element 2
- III. Element 3

Lösung 4

```
#set enum(numbering: "I.")  
+ Element 1  
+ Element 2  
+ Element 3
```

Aufgabe 5

Welcher Parameter kann durch eine set-Regel wie verändert werden, damit man eine nicht-nummerierte Listen wie folgt formatieren kann?
Die Dokumentation hilft!

- > Element 1
- > Element 2
- > Element 3

Lösung 5

```
#set list(marker: ">")
```

- Element 1
- Element 2
- Element 3

7. Eigene Templates und Skripts

7.1. Eigene Funktionen und Variablen #1

```
#let var = 3.14159
#let double(e) = {
    return 2*e
}

$pi$ ist etwa #var!
$tau$ ist etwa #double(var)!

$pi approx var$ \
$tau approx double(var)$
```

π ist etwa 3.14159!
 τ ist etwa 6.28318!

$\pi \approx 3.14159$
 $\tau \approx 6.28318$

7.2. Eigene Funktionen und Variablen #2

Eine Einschränkung: wir arbeiten mit *Pure Functions*.

Folgendes geht nicht:

```
#let var = 2
#let change_var(new_v) = {
    var = new_v
    return var
}
change_var(100)
```

Fehler:

```
#let change_var(new_v) = {
    var = new_v
    ^
variables from outside the
function are read-only and
cannot be modified
```

7.3. Eigene Funktionen und Variablen

#3

```
#let names = ("Peter", "Petra", "Josef", "Josefa")
#let greet(names) = {
    [Hallo ]
    names.join(last: " und ", ", ")
    [! #names.len() wundervolle Namen!]
}

#greet(names)
```

Hallo Peter, Petra, Josef und Josefa! 4 wundervolle Namen!

Aufgabe 6

Es soll eine Funktion erstellt werden, die zwei Zahlen addiert und das Ergebnis **rot** und **fett** formatiert. Etwa so:

```
#add(20, 40)
```

60

Lösung 6

Viele Möglichkeiten:

```
#let add(a, b) = strong(text(red, [$(a+b)]))  
#add(20, 40)
```

oder mit `#str()` Zahlen zu Strings machen:

```
#let add(a, b) = strong(text(red, str(a+b)))  
#add(20, 40)
```

oder mit Klammern:

```
#let add(a, b) = {  
  let result = a+b  
  strong(text(red, str(result)))  
}  
  
#add(20, 40)
```

8. Typst kann noch mehr!

8.1. Figuren und Referenzen¹

@glacier shows a glacier. Glaciers are complex systems.

```
#figure(  
  image("glacier.jpg", height: 80%),  
  caption: [A curious figure.],  
) <glacier>
```

Figure 1 shows a glacier. Glaciers are complex systems.



Figure 1: A curious figure.

8.2. Bibliographie¹

works.bib:

```
@article{netwok,
  title={At-scale impact of the {Net Wok}: A culinarily holistic investigation of distributed dumplings},
  author={Astley, Rick and Morris, Linda},
  journal={Armenian Journal of Proceedings},
  volume={61},
  pages={192--219},
  year={2020},
  publisher={Automattic Inc.}

}

@article{arrgh,
  title={The Pirate Organization},
  author={Leeson, Peter T.},
}
```

This was already noted by pirates long ago. @arrgh

Multiple sources say ...
`#cite("arrgh", "netwok").`

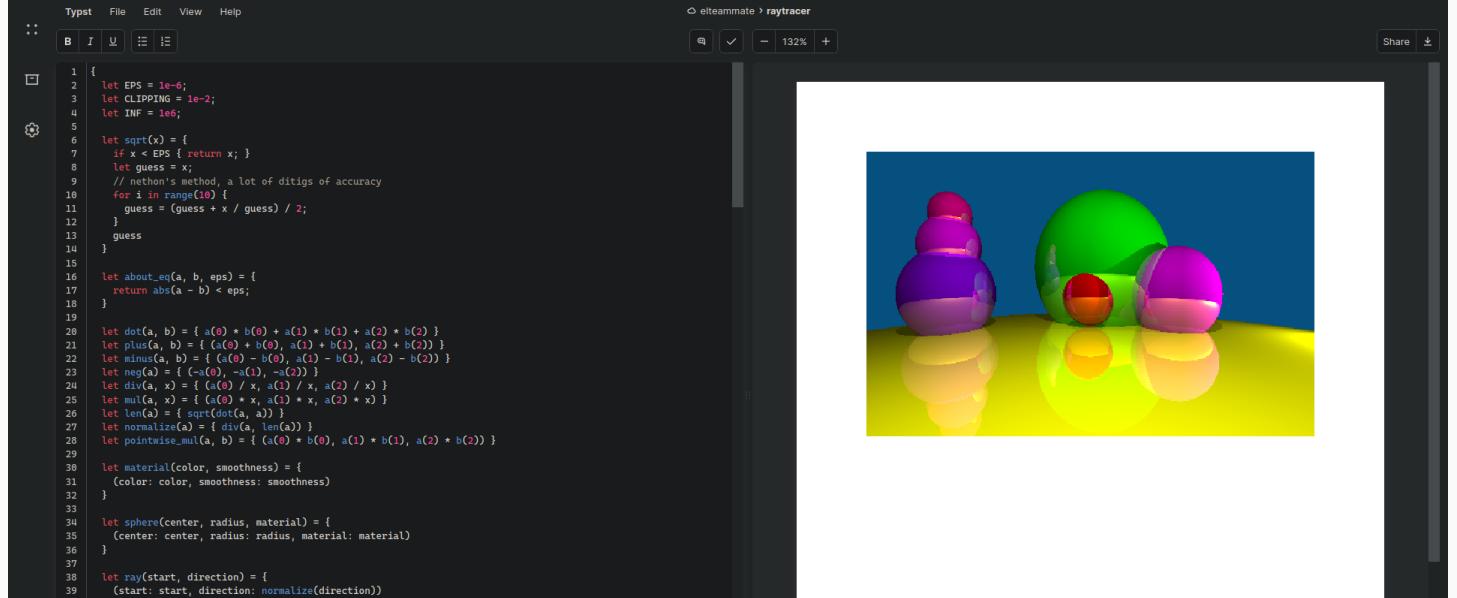
`#bibliography("works.bib")`

This was already noted by pirates long ago. [1]
Multiple sources say ... [1, 2].

Bibliography

- [1] P. T. Leeson, “The pirate organization.”
- [2] R. Astley, and L. Morris, “At-scale impact of the Net Wok: a culinarily holistic investigation of distributed dumplings,” *Armenian J. Proc.*, vol. 61, pp. 192–219, 2020.

8.3. Raytracing



The image shows a code editor window titled "raytracer.js" and a rendered 3D scene. The code editor displays a JavaScript file containing functions for vector operations, material definitions, sphere creation, and ray tracing. The rendered scene shows several spheres of different sizes and colors (green, purple, red, yellow) on a reflective surface against a blue background.

```
Typst  File  Edit  View  Help
B  I  U  ☰  ☱
1 {
2   let EPS = 1e-6;
3   let CLIPPING = 1e-2;
4   let INF = 1e6;
5
6   let sqrt(x) = {
7     if x < EPS { return x; }
8     let guess = x;
9     // newton's method, a lot of digits of accuracy
10    for i in range(10) {
11      guess = (guess + x / guess) / 2;
12    }
13    guess
14  }
15
16  let about_eq(a, b, eps) = {
17    return abs(a - b) < eps;
18  }
19
20  let dot(a, b) = { a(0) * b(0) + a(1) * b(1) + a(2) * b(2) }
21  let plus(a, b) = { (a(0) + b(0), a(1) + b(1), a(2) + b(2)) }
22  let minus(a, b) = { (a(0) - b(0), a(1) - b(1), a(2) - b(2)) }
23  let neg(a) = { -a(0), -a(1), -a(2) }
24  let div(a, x) = { (a(0) / x, a(1) / x, a(2) / x) }
25  let mul(a, x) = { (a(0) * x, a(1) * x, a(2) * x) }
26  let len(a) = { sqrt(dot(a, a)) }
27  let normalize(a) = { div(a, len(a)) }
28  let pointwise_mul(a, b) = { (a(0) * b(0), a(1) * b(1), a(2) * b(2)) }
29
30  let material(color, smoothness) = {
31    (color, smoothness)
32  }
33
34  let sphere(center, radius, material) = {
35    (center, radius, material)
36  }
37
38  let ray(start, direction) = [
39    (start, start, direction)
40  ]
41
42  let intersect(ray, spheres) = {
43    let ray_start = ray[0];
44    let ray_dir = ray[1];
45    let ray_end = ray_start + ray_dir * 1000;
46
47    let closest = null;
48    let closest_dist_sq = null;
49
50    for (let i = 0; i < spheres.length; i++) {
51      let sphere = spheres[i];
52
53      let center = sphere[0];
54      let radius = sphere[1];
55
56      let ray_center = ray_start + ray_dir * radius;
57
58      let dist_sq = distance_sq(ray_center, center);
59
60      if (dist_sq < radius * radius) {
61        if (closest == null || dist_sq < closest_dist_sq) {
62          closest = sphere;
63          closest_dist_sq = dist_sq;
64        }
65      }
66    }
67
68    closest
69  }
70
71  let distance_sq(a, b) = {
72    let dx = a(0) - b(0);
73    let dy = a(1) - b(1);
74    let dz = a(2) - b(2);
75
76    dx * dx + dy * dy + dz * dz
77  }
78
79  let color(ray, spheres) = {
80    let intersection = intersect(ray, spheres);
81
82    if (intersection == null) {
83      return [0, 0, 0];
84    }
85
86    let sphere = intersection[0];
87    let material = intersection[1];
88
89    let center = sphere[0];
90    let radius = sphere[1];
91
92    let ray_start = ray[0];
93    let ray_dir = ray[1];
94
95    let ray_center = ray_start + ray_dir * radius;
96
97    let normal = ray_center - center;
98    let normal_len = len(normal);
99
100   let normalized_normal = { x: normal(0) / normal_len,
101     y: normal(1) / normal_len,
102     z: normal(2) / normal_len
103   };
104
105   let eye = ray_center;
106   let light = [100, 100, 100];
107
108   let light_dot_normal = dot(light, normalized_normal);
109
110   let ambient = [0.1, 0.1, 0.1];
111
112   let diffuse = [
113     material.color.x * light_dot_normal,
114     material.color.y * light_dot_normal,
115     material.color.z * light_dot_normal
116   ];
117
118   let specular = [
119     material.smoothness * light_dot_normal * 1000,
120     material.smoothness * light_dot_normal * 1000,
121     material.smoothness * light_dot_normal * 1000
122   ];
123
124   let total_color = [
125     ambient.x + diffuse.x + specular.x,
126     ambient.y + diffuse.y + specular.y,
127     ambient.z + diffuse.z + specular.z
128   ];
129
130   total_color
131  }
132
133  let render(spheres) = {
134    let width = 1000;
135    let height = 1000;
136
137    let aspect_ratio = width / height;
138
139    let camera = { x: 0, y: 0, z: 10 };
140
141    let ray_start = camera;
142    let ray_dir = { x: 0, y: 0, z: 1 };
143
144    let rays = [];
145
146    for (let y = 0; y < height; y++) {
147      for (let x = 0; x < width; x++) {
148        let ray_dir_x = (x / width - 0.5) * 2;
149        let ray_dir_y = (y / height - 0.5) * 2;
150
151        let ray_dir_z = sqrt(1 - ray_dir_x * ray_dir_x - ray_dir_y * ray_dir_y);
152
153        let ray_dir_normalized = { x: ray_dir_x / ray_dir_z,
154          y: ray_dir_y / ray_dir_z,
155          z: 1 / ray_dir_z
156        };
157
158        let ray = [ray_start, ray_dir_normalized];
159
160        rays.push(ray);
161      }
162    }
163
164    let colors = rays.map(ray => color(ray, spheres));
165
166    let image = colors.map(colors => colors.map(color => color / 255));
167
168    image
169  }
170
171  let main() = {
172    let spheres = [
173      { center: [100, 100, 100], radius: 50, material: material([1, 0, 0], 100) },
174      { center: [200, 200, 200], radius: 30, material: material([0, 1, 0], 100) },
175      { center: [300, 300, 300], radius: 20, material: material([0, 0, 1], 100) },
176      { center: [400, 400, 400], radius: 10, material: material([1, 0, 1], 100) },
177      { center: [500, 500, 500], radius: 5, material: material([1, 1, 0], 100) }
178    ];
179
180    let image = render(spheres);
181
182    image
183  }
184
185  main()
186
187  export default main;
```

Voll funktionsfähiger Raytracer für 3D-Rendering.¹

9. Abschluss und Weiteres

9.1. Was noch fehlt¹

- Fußnoten (am 20.05.2023 mit v0.4.0 hinzugefügt)
- Paketmanager (kurzfristige Alternative: GitHub)
- StackOverflow (kurzfristige Alternative: Discord)

9.2. Erwartete Neuerungen¹

- die komplette Überarbeitung der Layout-Engine
- Paketmanager
- Verbesserung des Mathe-Layouts
- HTML Output
- <https://typst.app/docs/roadmap>

9.3. Wer sollte Typst (nicht) benutzen?

Pros:

- ✓ **steile** Lernkurve
- ✓ viele Programmierer-Ansätze, extrem einfach erweiterbar
- ✓ aktive Community
- ✓ schnelle Kompilierzeit
- ✓ online IDE
- ✓ verständliche Fehlermeldungen

Cons:

- ✗ viele Programmierer-Ansätze
- ✗ komplexes Layouting (keine Floating Figures)
- ✗ Pure Functions können schwer sein (States, Counter, ...)
- ✗ bisher keine Unterstützung durch große Journals
- ✗ kein zentrales Paketmanagement

(Stand: 25.05.2023)

9.4. Weiteres

Übrigens: Diese gesamte Präsentation wurde alleine in Typst erstellt.

- Typst Dokumentation: <https://typst.app/docs/>
- Offizielles Typst-Tutorial: <https://typst.app/docs/tutorial>
- Offizieller Typst-Discord: <https://discord.gg/2uDybryKPe>
- Code für diese Präsentation und weitere Beispiele: <https://github.com/survari/typst-seminar>
- Masterarbeit von Laurenz Mädje über Typst: <https://www.user.tu-berlin.de/laurmaedje/programmable-markup-language-for-typesetting.pdf>
- Lambdas, States und Counter: <https://typst.app/project/rpnqiqoQNfxXjHQmpJ81nF>
- Liste mit Typst-Projekten: <https://github.com/qjcg/awesome-typst>