

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-Fehlermeldung (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Rule 'pdflatex': The following rules & subrules became out-of-date: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/>

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?

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

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. `_#lorem(20)_`

```

\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)

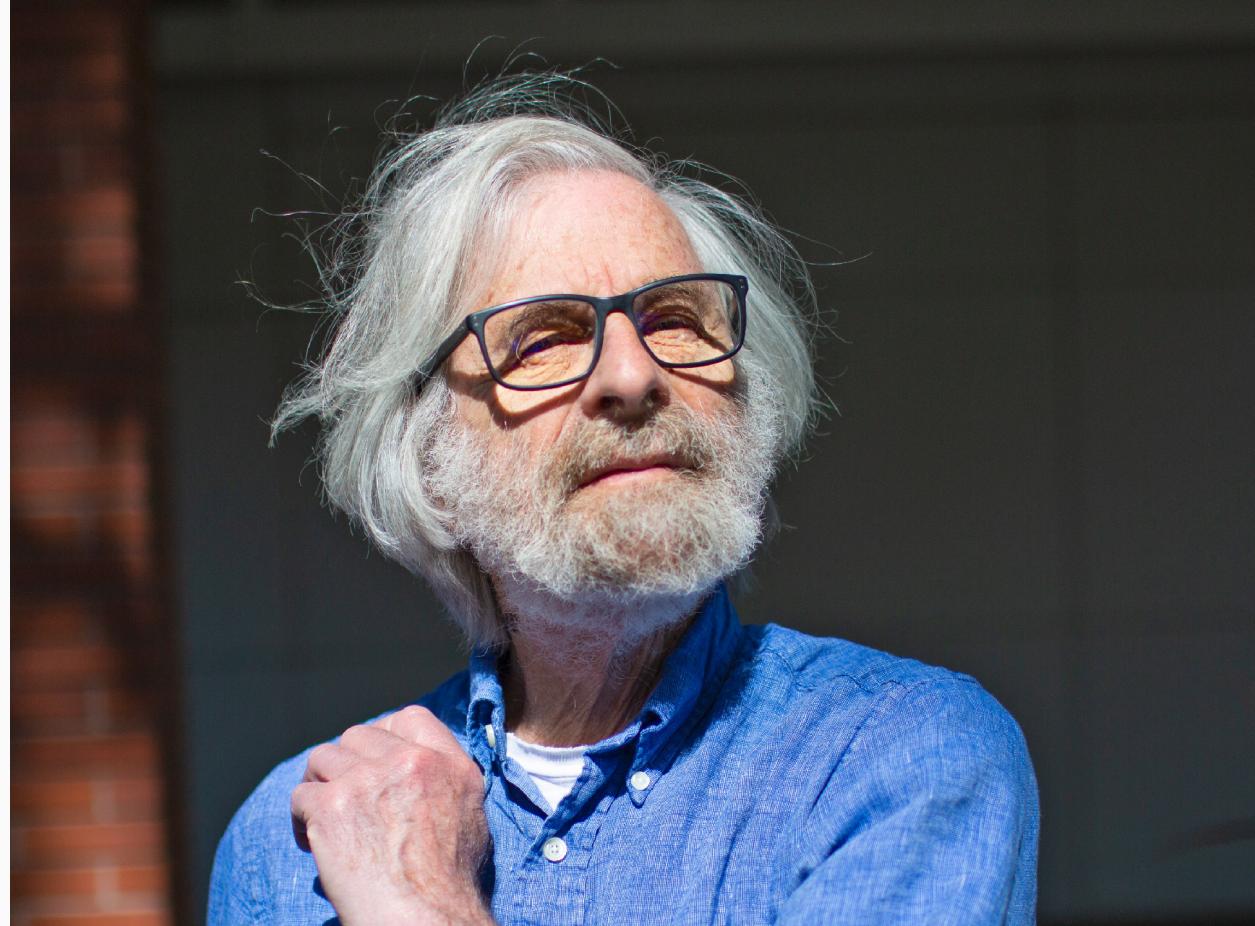
6. Die Typst-Dokumentation...

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

6.1. Dokumentations-Beispiel: image-Funktion

image kann noch ein bisschen was!

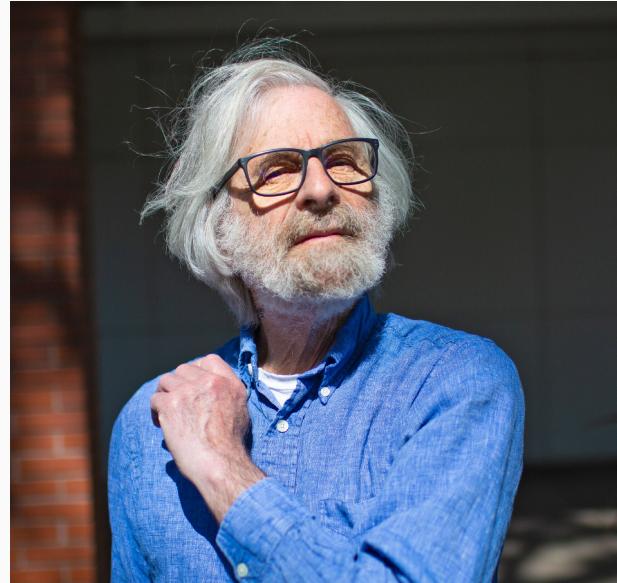
```
#image("leslie_lamport.png")
```



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").`

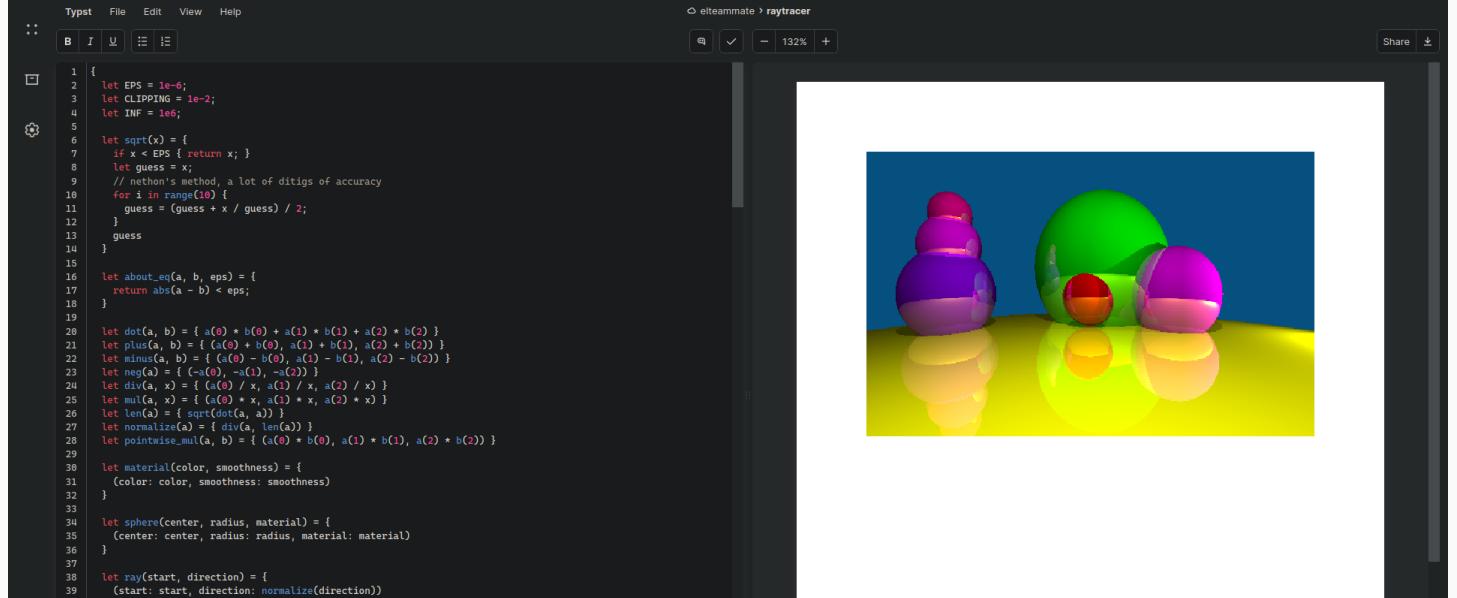
`#bibliography("works.bib")`

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 handling, 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 (start, end) = ray;
44    let closest = null;
45    let closestDist = INF;
46
47    for (let sphere of spheres) {
48      let center = sphere[0];
49      let radius = sphere[1];
50
51      let distance = dist(start, center);
52      if (distance <= radius) {
53        let rayDir = ray[1];
54        let rayStart = ray[0];
55
56        let t = (radius - distance) / rayDir;
57
58        if (t >= 0) {
59          closest = sphere;
60          closestDist = t;
61        }
62      }
63    }
64
65    if (closest) {
66      return closest;
67    } else {
68      return null;
69    }
70  }
71
72  let clip(ray, plane) = {
73    let (start, end) = ray;
74    let (normal, d) = plane;
75
76    let sign = dot(normal, start) + d;
77    if (sign <= 0) {
78      return ray;
79    }
80
81    let t = (-d - sign) / dot(normal, end);
82
83    let startClip = start + t * end;
84
85    return [start, startClip];
86  }
87
88  let intersectRaySphere(ray, sphere) = {
89    let (start, end) = ray;
90    let (center, radius) = sphere;
91
92    let distance = dist(start, center);
93    if (distance > radius) {
94      return null;
95    }
96
97    let rayDir = end - start;
98    let rayStart = start;
99
100   let t = ((radius - distance) * radius) / dot(rayDir, rayDir);
101
102   let intersection = rayStart + t * rayDir;
103
104   return intersection;
105 }
106
107 let intersectRayRay(ray1, ray2) = {
108   let (start1, end1) = ray1;
109   let (start2, end2) = ray2;
110
111   let (normal1, d1) = intersectRaySphere(ray1, spheres);
112   let (normal2, d2) = intersectRaySphere(ray2, spheres);
113
114   let sign1 = dot(normal1, start1) + d1;
115   let sign2 = dot(normal2, start2) + d2;
116
117   if (sign1 * sign2 < 0) {
118     let t1 = (-d1 - sign1) / dot(normal1, end1);
119     let t2 = (-d2 - sign2) / dot(normal2, end2);
120
121     if (t1 >= 0 & t2 >= 0) {
122       let intersection = start1 + t1 * end1;
123
124       return intersection;
125     }
126   }
127
128   return null;
129 }
130
131 let intersectRayPlane(ray, plane) = {
132   let (start, end) = ray;
133   let (normal, d) = plane;
134
135   let sign = dot(normal, start) + d;
136   if (sign <= 0) {
137     return ray;
138   }
139
140   let t = (-d - sign) / dot(normal, end);
141
142   let intersection = start + t * end;
143
144   return intersection;
145 }
146
147 let intersectRayRayClipped(ray1, ray2) = {
148   let (start1, end1) = ray1;
149   let (start2, end2) = ray2;
150
151   let (normal1, d1) = intersectRaySphere(ray1, spheres);
152   let (normal2, d2) = intersectRaySphere(ray2, spheres);
153
154   let sign1 = dot(normal1, start1) + d1;
155   let sign2 = dot(normal2, start2) + d2;
156
157   if (sign1 * sign2 < 0) {
158     let t1 = (-d1 - sign1) / dot(normal1, end1);
159     let t2 = (-d2 - sign2) / dot(normal2, end2);
160
161     if (t1 >= 0 & t2 >= 0) {
162       let intersection = start1 + t1 * end1;
163
164       return intersection;
165     }
166   }
167
168   return null;
169 }
170
171 let intersectRayRayClipped(ray1, ray2) = {
172   let (start1, end1) = ray1;
173   let (start2, end2) = ray2;
174
175   let (normal1, d1) = intersectRaySphere(ray1, spheres);
176   let (normal2, d2) = intersectRaySphere(ray2, spheres);
177
178   let sign1 = dot(normal1, start1) + d1;
179   let sign2 = dot(normal2, start2) + d2;
180
181   if (sign1 * sign2 < 0) {
182     let t1 = (-d1 - sign1) / dot(normal1, end1);
183     let t2 = (-d2 - sign2) / dot(normal2, end2);
184
185     if (t1 >= 0 & t2 >= 0) {
186       let intersection = start1 + t1 * end1;
187
188       return intersection;
189     }
190   }
191
192   return null;
193 }
194
195 let intersectRayRayClipped(ray1, ray2) = {
196   let (start1, end1) = ray1;
197   let (start2, end2) = ray2;
198
199   let (normal1, d1) = intersectRaySphere(ray1, spheres);
200   let (normal2, d2) = intersectRaySphere(ray2, spheres);
201
202   let sign1 = dot(normal1, start1) + d1;
203   let sign2 = dot(normal2, start2) + d2;
204
205   if (sign1 * sign2 < 0) {
206     let t1 = (-d1 - sign1) / dot(normal1, end1);
207     let t2 = (-d2 - sign2) / dot(normal2, end2);
208
209     if (t1 >= 0 & t2 >= 0) {
210       let intersection = start1 + t1 * end1;
211
212       return intersection;
213     }
214   }
215
216   return null;
217 }
218
219 let intersectRayRayClipped(ray1, ray2) = {
220   let (start1, end1) = ray1;
221   let (start2, end2) = ray2;
222
223   let (normal1, d1) = intersectRaySphere(ray1, spheres);
224   let (normal2, d2) = intersectRaySphere(ray2, spheres);
225
226   let sign1 = dot(normal1, start1) + d1;
227   let sign2 = dot(normal2, start2) + d2;
228
229   if (sign1 * sign2 < 0) {
230     let t1 = (-d1 - sign1) / dot(normal1, end1);
231     let t2 = (-d2 - sign2) / dot(normal2, end2);
232
233     if (t1 >= 0 & t2 >= 0) {
234       let intersection = start1 + t1 * end1;
235
236       return intersection;
237     }
238   }
239
240   return null;
241 }
242
243 let intersectRayRayClipped(ray1, ray2) = {
244   let (start1, end1) = ray1;
245   let (start2, end2) = ray2;
246
247   let (normal1, d1) = intersectRaySphere(ray1, spheres);
248   let (normal2, d2) = intersectRaySphere(ray2, spheres);
249
250   let sign1 = dot(normal1, start1) + d1;
251   let sign2 = dot(normal2, start2) + d2;
252
253   if (sign1 * sign2 < 0) {
254     let t1 = (-d1 - sign1) / dot(normal1, end1);
255     let t2 = (-d2 - sign2) / dot(normal2, end2);
256
257     if (t1 >= 0 & t2 >= 0) {
258       let intersection = start1 + t1 * end1;
259
260       return intersection;
261     }
262   }
263
264   return null;
265 }
266
267 let intersectRayRayClipped(ray1, ray2) = {
268   let (start1, end1) = ray1;
269   let (start2, end2) = ray2;
270
271   let (normal1, d1) = intersectRaySphere(ray1, spheres);
272   let (normal2, d2) = intersectRaySphere(ray2, spheres);
273
274   let sign1 = dot(normal1, start1) + d1;
275   let sign2 = dot(normal2, start2) + d2;
276
277   if (sign1 * sign2 < 0) {
278     let t1 = (-d1 - sign1) / dot(normal1, end1);
279     let t2 = (-d2 - sign2) / dot(normal2, end2);
280
281     if (t1 >= 0 & t2 >= 0) {
282       let intersection = start1 + t1 * end1;
283
284       return intersection;
285     }
286   }
287
288   return null;
289 }
290
291 let intersectRayRayClipped(ray1, ray2) = {
292   let (start1, end1) = ray1;
293   let (start2, end2) = ray2;
294
295   let (normal1, d1) = intersectRaySphere(ray1, spheres);
296   let (normal2, d2) = intersectRaySphere(ray2, spheres);
297
298   let sign1 = dot(normal1, start1) + d1;
299   let sign2 = dot(normal2, start2) + d2;
300
301   if (sign1 * sign2 < 0) {
302     let t1 = (-d1 - sign1) / dot(normal1, end1);
303     let t2 = (-d2 - sign2) / dot(normal2, end2);
304
305     if (t1 >= 0 & t2 >= 0) {
306       let intersection = start1 + t1 * end1;
307
308       return intersection;
309     }
310   }
311
312   return null;
313 }
314
315 let intersectRayRayClipped(ray1, ray2) = {
316   let (start1, end1) = ray1;
317   let (start2, end2) = ray2;
318
319   let (normal1, d1) = intersectRaySphere(ray1, spheres);
320   let (normal2, d2) = intersectRaySphere(ray2, spheres);
321
322   let sign1 = dot(normal1, start1) + d1;
323   let sign2 = dot(normal2, start2) + d2;
324
325   if (sign1 * sign2 < 0) {
326     let t1 = (-d1 - sign1) / dot(normal1, end1);
327     let t2 = (-d2 - sign2) / dot(normal2, end2);
328
329     if (t1 >= 0 & t2 >= 0) {
330       let intersection = start1 + t1 * end1;
331
332       return intersection;
333     }
334   }
335
336   return null;
337 }
338
339 let intersectRayRayClipped(ray1, ray2) = {
340   let (start1, end1) = ray1;
341   let (start2, end2) = ray2;
342
343   let (normal1, d1) = intersectRaySphere(ray1, spheres);
344   let (normal2, d2) = intersectRaySphere(ray2, spheres);
345
346   let sign1 = dot(normal1, start1) + d1;
347   let sign2 = dot(normal2, start2) + d2;
348
349   if (sign1 * sign2 < 0) {
350     let t1 = (-d1 - sign1) / dot(normal1, end1);
351     let t2 = (-d2 - sign2) / dot(normal2, end2);
352
353     if (t1 >= 0 & t2 >= 0) {
354       let intersection = start1 + t1 * end1;
355
356       return intersection;
357     }
358   }
359
360   return null;
361 }
362
363 let intersectRayRayClipped(ray1, ray2) = {
364   let (start1, end1) = ray1;
365   let (start2, end2) = ray2;
366
367   let (normal1, d1) = intersectRaySphere(ray1, spheres);
368   let (normal2, d2) = intersectRaySphere(ray2, spheres);
369
370   let sign1 = dot(normal1, start1) + d1;
371   let sign2 = dot(normal2, start2) + d2;
372
373   if (sign1 * sign2 < 0) {
374     let t1 = (-d1 - sign1) / dot(normal1, end1);
375     let t2 = (-d2 - sign2) / dot(normal2, end2);
376
377     if (t1 >= 0 & t2 >= 0) {
378       let intersection = start1 + t1 * end1;
379
380       return intersection;
381     }
382   }
383
384   return null;
385 }
386
387 let intersectRayRayClipped(ray1, ray2) = {
388   let (start1, end1) = ray1;
389   let (start2, end2) = ray2;
390
391   let (normal1, d1) = intersectRaySphere(ray1, spheres);
392   let (normal2, d2) = intersectRaySphere(ray2, spheres);
393
394   let sign1 = dot(normal1, start1) + d1;
395   let sign2 = dot(normal2, start2) + d2;
396
397   if (sign1 * sign2 < 0) {
398     let t1 = (-d1 - sign1) / dot(normal1, end1);
399     let t2 = (-d2 - sign2) / dot(normal2, end2);
400
401     if (t1 >= 0 & t2 >= 0) {
402       let intersection = start1 + t1 * end1;
403
404       return intersection;
405     }
406   }
407
408   return null;
409 }
410
411 let intersectRayRayClipped(ray1, ray2) = {
412   let (start1, end1) = ray1;
413   let (start2, end2) = ray2;
414
415   let (normal1, d1) = intersectRaySphere(ray1, spheres);
416   let (normal2, d2) = intersectRaySphere(ray2, spheres);
417
418   let sign1 = dot(normal1, start1) + d1;
419   let sign2 = dot(normal2, start2) + d2;
420
421   if (sign1 * sign2 < 0) {
422     let t1 = (-d1 - sign1) / dot(normal1, end1);
423     let t2 = (-d2 - sign2) / dot(normal2, end2);
424
425     if (t1 >= 0 & t2 >= 0) {
426       let intersection = start1 + t1 * end1;
427
428       return intersection;
429     }
430   }
431
432   return null;
433 }
434
435 let intersectRayRayClipped(ray1, ray2) = {
436   let (start1, end1) = ray1;
437   let (start2, end2) = ray2;
438
439   let (normal1, d1) = intersectRaySphere(ray1, spheres);
440   let (normal2, d2) = intersectRaySphere(ray2, spheres);
441
442   let sign1 = dot(normal1, start1) + d1;
443   let sign2 = dot(normal2, start2) + d2;
444
445   if (sign1 * sign2 < 0) {
446     let t1 = (-d1 - sign1) / dot(normal1, end1);
447     let t2 = (-d2 - sign2) / dot(normal2, end2);
448
449     if (t1 >= 0 & t2 >= 0) {
450       let intersection = start1 + t1 * end1;
451
452       return intersection;
453     }
454   }
455
456   return null;
457 }
458
459 let intersectRayRayClipped(ray1, ray2) = {
460   let (start1, end1) = ray1;
461   let (start2, end2) = ray2;
462
463   let (normal1, d1) = intersectRaySphere(ray1, spheres);
464   let (normal2, d2) = intersectRaySphere(ray2, spheres);
465
466   let sign1 = dot(normal1, start1) + d1;
467   let sign2 = dot(normal2, start2) + d2;
468
469   if (sign1 * sign2 < 0) {
470     let t1 = (-d1 - sign1) / dot(normal1, end1);
471     let t2 = (-d2 - sign2) / dot(normal2, end2);
472
473     if (t1 >= 0 & t2 >= 0) {
474       let intersection = start1 + t1 * end1;
475
476       return intersection;
477     }
478   }
479
480   return null;
481 }
482
483 let intersectRayRayClipped(ray1, ray2) = {
484   let (start1, end1) = ray1;
485   let (start2, end2) = ray2;
486
487   let (normal1, d1) = intersectRaySphere(ray1, spheres);
488   let (normal2, d2) = intersectRaySphere(ray2, spheres);
489
490   let sign1 = dot(normal1, start1) + d1;
491   let sign2 = dot(normal2, start2) + d2;
492
493   if (sign1 * sign2 < 0) {
494     let t1 = (-d1 - sign1) / dot(normal1, end1);
495     let t2 = (-d2 - sign2) / dot(normal2, end2);
496
497     if (t1 >= 0 & t2 >= 0) {
498       let intersection = start1 + t1 * end1;
499
500       return intersection;
501     }
502   }
503
504   return null;
505 }
506
507 let intersectRayRayClipped(ray1, ray2) = {
508   let (start1, end1) = ray1;
509   let (start2, end2) = ray2;
510
511   let (normal1, d1) = intersectRaySphere(ray1, spheres);
512   let (normal2, d2) = intersectRaySphere(ray2, spheres);
513
514   let sign1 = dot(normal1, start1) + d1;
515   let sign2 = dot(normal2, start2) + d2;
516
517   if (sign1 * sign2 < 0) {
518     let t1 = (-d1 - sign1) / dot(normal1, end1);
519     let t2 = (-d2 - sign2) / dot(normal2, end2);
520
521     if (t1 >= 0 & t2 >= 0) {
522       let intersection = start1 + t1 * end1;
523
524       return intersection;
525     }
526   }
527
528   return null;
529 }
530
531 let intersectRayRayClipped(ray1, ray2) = {
532   let (start1, end1) = ray1;
533   let (start2, end2) = ray2;
534
535   let (normal1, d1) = intersectRaySphere(ray1, spheres);
536   let (normal2, d2) = intersectRaySphere(ray2, spheres);
537
538   let sign1 = dot(normal1, start1) + d1;
539   let sign2 = dot(normal2, start2) + d2;
540
541   if (sign1 * sign2 < 0) {
542     let t1 = (-d1 - sign1) / dot(normal1, end1);
543     let t2 = (-d2 - sign2) / dot(normal2, end2);
544
545     if (t1 >= 0 & t2 >= 0) {
546       let intersection = start1 + t1 * end1;
547
548       return intersection;
549     }
550   }
551
552   return null;
553 }
554
555 let intersectRayRayClipped(ray1, ray2) = {
556   let (start1, end1) = ray1;
557   let (start2, end2) = ray2;
558
559   let (normal1, d1) = intersectRaySphere(ray1, spheres);
560   let (normal2, d2) = intersectRaySphere(ray2, spheres);
561
562   let sign1 = dot(normal1, start1) + d1;
563   let sign2 = dot(normal2, start2) + d2;
564
565   if (sign1 * sign2 < 0) {
566     let t1 = (-d1 - sign1) / dot(normal1, end1);
567     let t2 = (-d2 - sign2) / dot(normal2, end2);
568
569     if (t1 >= 0 & t2 >= 0) {
570       let intersection = start1 + t1 * end1;
571
572       return intersection;
573     }
574   }
575
576   return null;
577 }
578
579 let intersectRayRayClipped(ray1, ray2) = {
580   let (start1, end1) = ray1;
581   let (start2, end2) = ray2;
582
583   let (normal1, d1) = intersectRaySphere(ray1, spheres);
584   let (normal2, d2) = intersectRaySphere(ray2, spheres);
585
586   let sign1 = dot(normal1, start1) + d1;
587   let sign2 = dot(normal2, start2) + d2;
588
589   if (sign1 * sign2 < 0) {
590     let t1 = (-d1 - sign1) / dot(normal1, end1);
591     let t2 = (-d2 - sign2) / dot(normal2, end2);
592
593     if (t1 >= 0 & t2 >= 0) {
594       let intersection = start1 + t1 * end1;
595
596       return intersection;
597     }
598   }
599
600   return null;
601 }
602
603 let intersectRayRayClipped(ray1, ray2) = {
604   let (start1, end1) = ray1;
605   let (start2, end2) = ray2;
606
607   let (normal1, d1) = intersectRaySphere(ray1, spheres);
608   let (normal2, d2) = intersectRaySphere(ray2, spheres);
609
610   let sign1 = dot(normal1, start1) + d1;
611   let sign2 = dot(normal2, start2) + d2;
612
613   if (sign1 * sign2 < 0) {
614     let t1 = (-d1 - sign1) / dot(normal1, end1);
615     let t2 = (-d2 - sign2) / dot(normal2, end2);
616
617     if (t1 >= 0 & t2 >= 0) {
618       let intersection = start1 + t1 * end1;
619
620       return intersection;
621     }
622   }
623
624   return null;
625 }
626
627 let intersectRayRayClipped(ray1, ray2) = {
628   let (start1, end1) = ray1;
629   let (start2, end2) = ray2;
630
631   let (normal1, d1) = intersectRaySphere(ray1, spheres);
632   let (normal2, d2) = intersectRaySphere(ray2, spheres);
633
634   let sign1 = dot(normal1, start1) + d1;
635   let sign2 = dot(normal2, start2) + d2;
636
637   if (sign1 * sign2 < 0) {
638     let t1 = (-d1 - sign1) / dot(normal1, end1);
639     let t2 = (-d2 - sign2) / dot(normal2, end2);
640
641     if (t1 >= 0 & t2 >= 0) {
642       let intersection = start1 + t1 * end1;
643
644       return intersection;
645     }
646   }
647
648   return null;
649 }
650
651 let intersectRayRayClipped(ray1, ray2) = {
652   let (start1, end1) = ray1;
653   let (start2, end2) = ray2;
654
655   let (normal1, d1) = intersectRaySphere(ray1, spheres);
656   let (normal2, d2) = intersectRaySphere(ray2, spheres);
657
658   let sign1 = dot(normal1, start1) + d1;
659   let sign2 = dot(normal2, start2) + d2;
660
661   if (sign1 * sign2 < 0) {
662     let t1 = (-d1 - sign1) / dot(normal1, end1);
663     let t2 = (-d2 - sign2) / dot(normal2, end2);
664
665     if (t1 >= 0 & t2 >= 0) {
666       let intersection = start1 + t1 * end1;
667
668       return intersection;
669     }
670   }
671
672   return null;
673 }
674
675 let intersectRayRayClipped(ray1, ray2) = {
676   let (start1, end1) = ray1;
677   let (start2, end2) = ray2;
678
679   let (normal1, d1) = intersectRaySphere(ray1, spheres);
680   let (normal2, d2) = intersectRaySphere(ray2, spheres);
681
682   let sign1 = dot(normal1, start1) + d1;
683   let sign2 = dot(normal2, start2) + d2;
684
685   if (sign1 * sign2 < 0) {
686     let t1 = (-d1 - sign1) / dot(normal1, end1);
687     let t2 = (-d2 - sign2) / dot(normal2, end2);
688
689     if (t1 >= 0 & t2 >= 0) {
690       let intersection = start1 + t1 * end1;
691
692       return intersection;
693     }
694   }
695
696   return null;
697 }
698
699 let intersectRayRayClipped(ray1, ray2) = {
700   let (start1, end1) = ray1;
701   let (start2, end2) = ray2;
702
703   let (normal1, d1) = intersectRaySphere(ray1, spheres);
704   let (normal2, d2) = intersectRaySphere(ray2, spheres);
705
706   let sign1 = dot(normal1, start1) + d1;
707   let sign2 = dot(normal2, start2) + d2;
708
709   if (sign1 * sign2 < 0) {
710     let t1 = (-d1 - sign1) / dot(normal1, end1);
711     let t2 = (-d2 - sign2) / dot(normal2, end2);
712
713     if (t1 >= 0 & t2 >= 0) {
714       let intersection = start1 + t1 * end1;
715
716       return intersection;
717     }
718   }
719
720   return null;
721 }
722
723 let intersectRayRayClipped(ray1, ray2) = {
724   let (start1, end1) = ray1;
725   let (start2, end2) = ray2;
726
727   let (normal1, d1) = intersectRaySphere(ray1, spheres);
728   let (normal2, d2) = intersectRaySphere(ray2, spheres);
729
730   let sign1 = dot(normal1, start1) + d1;
731   let sign2 = dot(normal2, start2) + d2;
732
733   if (sign1 * sign2 < 0) {
734     let t1 = (-d1 - sign1) / dot(normal1, end1);
735     let t2 = (-d2 - sign2) / dot(normal2, end2);
736
737     if (t1 >= 0 & t2 >= 0) {
738       let intersection = start1 + t1 * end1;
739
740       return intersection;
741     }
742   }
743
744   return null;
745 }
746
747 let intersectRayRayClipped(ray1, ray2) = {
748   let (start1, end1) = ray1;
749   let (start2, end2) = ray2;
750
751   let (normal1, d1) = intersectRaySphere(ray1, spheres);
752   let (normal2, d2) = intersectRaySphere(ray2, spheres);
753
754   let sign1 = dot(normal1, start1) + d1;
755   let sign2 = dot(normal2, start2) + d2;
756
757   if (sign1 * sign2 < 0) {
758     let t1 = (-d1 - sign1) / dot(normal1, end1);
759     let t2 = (-d2 - sign2) / dot(normal2, end2);
760
761     if (t1 >= 0 & t2 >= 0) {
762       let intersection = start1 + t1 * end1;
763
764       return intersection;
765     }
766   }
767
768 }
```

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
- ...

9.3. Wer sollte Typst (nicht) benutzen?

Pros:

- ✓ **steile** Lernkurve
- ✓ viele Programmierer-Ansätze
- ✓ sehr dynamisch
- ✓ aktive Community
- ✓ schnelle Kompilierzeit
- ✓ 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>