Das schule Paket

Hilfen für den Schulalltag

v1.0.1 2025-12-15 MIT

Vorlagen und Makros für die Schule.

J. Neugebauer

https://github.com/jneug

ngb.schule

schule ist eine Sammlung von Typst Vorlagen zur Gestaltung von Arbeitsmaterialien (Arbeitsblätter, Klausuren, Wochenpläne ...) für die Schule. Das Paket ist eine Adaption meines LaTeX Pakets[1] für Typst.

Table of Contents

[1] https://github.com/jneug/arbeitsblatt

Part I

Über das Paket

SCHULE ist ein Typst Paket, mit dem Arbeitsblätter, Klausuren und andere Materialien für den Informatikunterricht erstellt werden können.

Das Paket ist ein Port des LaTeX Pakets jneug/arbeitsblatt[2], welches wiederum auf dem schule Paket[3] basierte.

Durch die Adaption für Typst wurden mit der Zeit Funktionalitäten ergänzt, entfernt und verändert, um das Paket an die modernere Typst-Umgebung und das wachsende Ökosystem von verfügbaren Paketen anzupassen.

[2] https://github.com/jneug/arbeitsblatt
[3] https://ctan.org/pkg/schule

Part II

Verwendung der Vorlagen

II.1 Installation

Eine manuelle Installation ist nicht notwendig. Das Paket wird automatisch installiert, wenn es in einem Dokument verwendet wird. Dazu wird es einfach im Dokument importiert:

```
#import "@preview/schule:1.0.1": ab
#import ab: *
```

II.2 Vorlagen

Das Kernstück des Pakets sind die verschiedenen Vorlagen für verschiedene Arten von Dokumente. Dazu gehören Arbeitsblätter, Klassenarbeiten und Präsentationen.

II.2.1 Arbeitsblatt (ab)

II.2.2 Klassenarbeit (ka)

II.2.3 Klausur (kl)

II.2.4 Präsentation (pt)

II.2.5 Lerntheke (lt)

II.3 Module

II.3.1 Aufgaben

Ein zentrales Modul des Pakets sind `aufgaben`.

```
#vari
```

`#vari(..(args))`

Setzt eine Aufgabe in mehreren Varianten.

II.3.2 Layout- und Text-Elemente

#anh	#enumnn	#programm
#anhang	#hinweis	#quelle-neu
#aufg-neu	#infobox	#rahmen
#code	#kasten	#schattenbox
#container	#luecke	#sourcecode
#datei	#name	#taste
#enuma	#operator	#tastenkuerzel
#enumn	#ordner	#warnungbox

#anhang(..⟨page-args⟩) → `content`

Setzt den Inhalt als Anhang für das Dokument. Die Kopfzeile und die Numerierung der Überschriften wird angepasst.

Der Anhang wird als neuer Seitenabschnitt erstellt und kann daher im Format vom Hauptteil abweichen. Die Argumente werden direkt an #page weitergegeben.

Beispielsweise lässt sich der Anhang so im Querformat in zwei Spalten setzen:

```
#anhang(flipped: true, columns: 2)[
 #lorem(500)
]
```

Argument

..⟨page-args⟩ `any`

Alle Argumente werden an #page weitergegeben, um das Seitenformat des Anhangs zu ändern.

#anh(⟨label⟩, ⟨supplement⟩: "Anhang", ⟨prefix⟩: "anh:") → `content`

Erzeugt eine Referenz zu einem Abschnitt im Anhang.

```
#anh("doku")
```

Anhang A

Argument

⟨label⟩ `label` | `str`

Ziel der Referenz.

Argument

⟨supplement⟩: "Anhang" `str`

Ergänzendes Etikett.

Argument

⟨prefix⟩: "anh:" `str`

Prefix für Anhang-Labels.

#operator(⟨body⟩) → `content`

Auszeichnung von Operatoren:

```
1 #operator[Entwirf] einen Algorithmus und #operator[stelle] ihn
  in geeigneter Form #operator[dar].
2
3 #operator[Implementiere] den Algorithmus nach deinem Entwurf.
```

Entwirf einen Algorithmus und stelle ihn in geeigneter Form dar.

Implementiere den Algorithmus nach deinem Entwurf.

Argument

⟨body⟩ `str` | `content`

Operator zum hervorheben.

#name(⟨name⟩, ⟨last⟩: none) → `content`

Darstellung eines Namens:

- #name[Jonas Neugebauer] → Jonas Neugebauer
- #name[John William Mauchly] → John William Mauchly
- #name[Adriaan van Wijngaarden] → Adriaan van Wijngaarden
- #name("Adriaan", last:"van Wijngaarden") → Adriaan van Wijngaarden
- #name(last:2)[Adriaan van Wijngaarden] → Adriaan van Wijngaarden

Argument

⟨name⟩ `str` | `content`

Name, der dargestellt werden soll.

Argument

⟨last⟩: none — str | int

Optionaler Nachname, falls dieser aus mehreren Teilen besteht. Falls dies ein int ist, werden die letzten ⟨last⟩ Namensteile von ⟨name⟩ als Nachname verwendet.

#taste(⟨label⟩) → content

Formatierung von Tasten.

- #taste("Enter") → Enter
- #taste(sym.arrow.l.hook) → ↩

Argument

⟨label⟩ — str | content

Aufschrift der Taste.

#tastenkuerzel(..⟨labels⟩, ⟨sep⟩: box(inset: (x: .1em), "+")) → content

Formatierung von Tastenkürzeln.

- #tastenkuerzel("Strg","C") → Strg + C
- #tastenkuerzel("Strg","Cmd") / Strg + Cmd
- #tastenkuerzel("Strg","Shift","C", sep:sym.plus.o) → Strg ⊕ Shift ⊕ C

Argument

..⟨labels⟩ — str | content

Aufschriften der Tasten.

Argument

⟨sep⟩: box(inset: (x: .1em), "+") — str | content

Trennzeichen zwischen Tasten.

#datei(⟨name⟩) → content

Formatierung von Dateinamen.

- #datei("beispiel.typ") → beispiel.typ

Argument

⟨name⟩ — str | content

Name der Datei.

#ordner(⟨name⟩) → content

Formatierung von Ordnernamen.

- `#ordner("arbeitsblaetter")` → 🗀 arbeitsblaetter

Argument

⟨name⟩ `str` | `content`

Name des Ordners.

`#programm(⟨name⟩)` → `content`

Formatierung von Programmnamen.

- `#programm("VSCode")` → VSCode

Argument

⟨name⟩ `str` | `content`

Name des Programms.

`#aufg-neu(⟨prefix⟩, ⟨sep⟩: h(0.166667em), ⟨range-sep⟩: ", ")` → `function`

Erstellt ein neues Format zur Auflistung von Aufgabennummern. Die erstelle Funktion fasst ihre Argumente automatisch zu Bereichen zusammen und gibt sie möglichst kompakt aus.

```
#let nr = aufg-neu("Nr.")

- #nr(1,)
- #nr(1,3)
- #nr(1,2,3)
- #nr(1,2,3,6,7,8)
- #nr(1,2,3,6,7,8,12)
```

- Nr. 1
- Nr. 1 und 3
- Nr. 1–3
- Nr. 1–3 und 6–8
- Nr. 1–3, 6–8 und 12

Argument

⟨prefix⟩ `str` | `content`

Prefix für die Angabe.

Argument

⟨sep⟩: h(0.166667em) str | content

Trennzeichen für Prefix und Nummernbereiche.

Argument

⟨range-sep⟩: ", " str | content

Trennzeichen zwischen Nummernbereichen.

#quelle-neu(⟨name⟩, ⟨seiten-format⟩: seiten, ⟨aufgaben-format⟩: aufg, ⟨sep⟩: ": ") → function

Erstellt eine neue Funktion für Quellenangaben, die sich aus Seiten- und optionalen Aufgabennummern zusammensetzen.

```
#let pp = quelle-neu(
  "Präsentation",
  seiten-format: aufg-neu("Folie", sep: " "),
  sep: ", "
)

- #pp(4,3,2)
- #pp(79, (1,2,3))
- #pp((101,102), 16)
- #pp((101,102),       range(16, 19))
- #pp((101,102), 13,   range(16, 19))
- #pp((101,102), 13, ..range(16, 19))
- #pp( 101,102 , 13, ..range(16, 19))
- #pp( 101,102 , 13,   range(16, 19))
```

- Präsentation Folie 2–4
- Präsentation Folie 79, Aufg. 1–3
- Präsentation Folie 101 und 102, Aufg. 16
- Präsentation Folie 101 und 102, Aufg. 16–18
- Präsentation Folie 101 und 102, Aufg. 13 und 16–18
- Präsentation Folie 101 und 102, Aufg. 13 und 16–18
- Präsentation Folie 13, 16–18, 101 und 102
- Präsentation Folie 13, 101 und 102, Aufg. 16–18

Die erstellte Funktion

#luecke(⟨width⟩: 4cm, ⟨stroke⟩: 1pt + black, ⟨offset⟩: 2pt, ⟨text⟩: none) → **content**

Textlücken.

- `#luecke()` → _____
- `#luecke(width: 2cm, offset: 5pt)` → _____
- `#luecke(text: "Hallo Welt!", stroke: .5pt+red)` → Hallo Welt!

- width (length): Breite der Textlücke, wenn `⟨text⟩` nicht gegeben ist.
- text (length): Text, anhand dessen die Breite der Textlücke bestimmt werden soll. Falls angegeben, wird `⟨width⟩` ignoriert.
- stroke (length): Linienstil der Unterstreichung.
- offset (length): Abstand der Linie zur Basislinie des umliegenden Textes.

```
#container(
  ⟨width⟩: 100%,
  ⟨stroke⟩: 2pt + black,
  ⟨fill⟩: white,
  ⟨inset⟩: .64em,
  ⟨shadow⟩: 0pt,
  ⟨radius⟩: 3pt,
  ..⟨box-args⟩,
  ⟨body⟩
) → content
```

Eine generische Box um Inhalte. Verwendet SHOWYBOX[4]. Im Allgemeinen werden die spezifischeren Boxen benutzt:

- `#rahmen`
- `#kasten`
- `#schattenbox`
- `#infobox`
- `#warnungbox`

- width (length): Breite der Box.
- stroke (stroke): Rahmenlinie um die Box.
- fill (color): Hintergrundfarbe der Box.
- inset (length, dictionary): Innenabstände der Box.
- shadow (length): Schattenabstand.
- radius (length): Radius der abgrundeten Ecken.
- ..box-args (any): Weitere Argumente, die an SHOWYBOX weitergereicht werden.

[4] https://typst.app/universe/package/Showybox

```
#rahmen(
  (width): 100%,
  (stroke): 2pt + theme.secondary,
  (fill): white,
  (inset): .64em,
  (shadow): 0pt,
  (radius): 3pt,
  ..(box-args),
  (body)
) → content
```

Ein Rahmen um Inhalte.

```
1  #rahmen[#lorem(10)]
```

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do.

- ..args (any): Argumente für `#container`.

`#kasten(..(args)) → content`

Ein Kasten um Inhalte.

```
1  #kasten[#lorem(10)]
```

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do.

- ..args (any): Argumente für `#container`.

`#schattenbox(..(args)) → content`

Eine Box mit Schatten um Inhalte.

```
1  #schattenbox[#lorem(10)]
```

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do.

- ..args (any): Argumente für `#container`.

`#infobox(..(args), (body)) → content`

Eine Infobox um Inhalte.

```
1  #infobox[#lorem(10)]
```

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do.

- ..args (any): Argumente für `#container`.

`#warnungbox(..(args), (body))` → **`content`**

Eine Warnungsbox um Inhalte.

```
1  #warnungbox[#lorem(10)]
```

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do.

- ..args (any): Argumente für `#container`.

`#hinweis((typ): "Hinweis", (icon): icon("information-circle"), ..(clue-args), (body))` → **`content`**

Zeigt einen hervorgehobenen Hinweis an.

```
1  #pad(left:10pt, hinweis[#lorem(8)])
2
3  #hinweis(typ:"Tipp", icon:emoji.face.halo)[#lorem(8)]
```

Hinweis: Lorem ipsum dolor sit amet, consectetur adipiscing elit.

Tipp: Lorem ipsum dolor sit amet, consectetur adipiscing elit.

- typ (string): Art des Hinweises.
- icon (symbol): Ein Symbol für den Hinweis..
- body (content): Inhalte des Hinweises.

`#sourcecode(..(args))` → **`content`**

Zeigt Quelltext mit Zeilennummern und in einem `#frame` an. Alias für `#codelst.sourcecode`.

````
1 #sourcecode[```python
2 print("Hello, World!")
3 ```]
````

---

python

```
1 print("Hello, World!")
```

- ..args (any): Argument für `#codelst.sourcecode`.

`#code(⟨body⟩, ⟨lang⟩: none)` → **`content`**

Inline-Code mit Syntax-Highlighting. Im Prinzip gleichwertig mit der Auszeichnungsvariante mit drei Backticks:

- `#code(lang:"python", "print('Hallo, Welt')")` → `print('Hallo, Welt')`
- ```` ```python print('Hallo, Welt')``` ```` → `print('Hallo, Welt')`

`#enuma(⟨body⟩)` → **`content`**

Setzt das Nummernformat für Aufzählungen im `⟨body⟩` auf `a)`.

```
1 #enuma[
2   + Eins
3   + Zwei
4   + Drei
5 ]
```

---

a) Eins
b) Zwei
c) Drei

- body (content): Inhalte mit Aufzählungen.

`#enumn(⟨body⟩)` → **`content`**

Setzt das Nummernformat für Aufzählungen im `⟨body⟩` auf `1)`.

```
1 #enumn[
2   + Eins
3   + Zwei
4   + Drei
5 ]
```

---

1) Eins
2) Zwei
3) Drei

- body (content): Inhalte mit Aufzählungen.

`#enumnn((body))` → **`content`**

Setzt das Nummernformat für Aufzählungen im `⟨body⟩` auf `(1)`.

```
1 #enumnn[
2   + Eins
3   + Zwei
4   + Drei
5 ]
```

---

(1) Eins
(2) Zwei
(3) Drei

- body (content): Inhalte mit Aufzählungen.

`#icon`

Symbols

`#seiten` **`content`**

Seitennummern

- `#seiten(1,2,3,6,7,8,12)` → S. 1 – 3, 6 – 8 und 12

`#aufg` **`content`**

Aufgabennummern

- `#aufg(1,2,3,6,7,8,12)` → Aufg. 1 – 3, 6 – 8 und 12

```
#degr                #num                #small
#eur                 #perc               #squiggly
#highlight           #scaled             #unit
#large               #si                 #uunderline
```

#scaled((content), (factor): 0.8) → content

Skalierter text.

- `#scaled[Hallo Welt.]` → Hallo Welt.
- `#scaled(factor:.5)[Hallo Welt.]` → Hallo Welt.
- `#scaled("Hallo Welt.", factor:120%)` → Hallo Welt.

Argument

`⟨content⟩` str | content

Zu skalierender Text.

Argument

`⟨factor⟩: 0.8`

Der Skalierungsfaktor. float | ratio

#small((content)) → content

Kleiner text.

- `#small[#lorem(5)]` → Lorem ipsum dolor sit amet.
- `#small(lorem(5))` → Lorem ipsum dolor sit amet.

Argument

`⟨content⟩` str | content

Zu skalierender Text.

#large((content)) → content

Großer text.

- `#large[#lorem(5)]` → Lorem ipsum dolor sit amet.
- `#large(lorem(5))` → Lorem ipsum dolor sit amet.

Argument

`⟨content⟩` str | content

Zu skalierender Text.

```
#uunderline(
  ⟨stroke⟩: auto,
  ⟨offset⟩: auto,
  ⟨distance⟩: auto,
  ⟨extent⟩: 0pt,
  ⟨evade⟩: true,
  ⟨body⟩
) → content
```

Doppelte Unterstreichung.

- `#uunderline[#lorem(5)]` → Lorem ipsum dolor sit amet.
- `#uunderline(lorem(5), stroke:2pt+red, offset:2pt, distance: 1pt)` → Lorem ipsum dolor sit amet.

Argument

⟨stroke⟩: auto — stroke

Linienstil für die Unterstreichung.

Argument

⟨offset⟩: auto — length

Abstand der oberen Linie zum Text.

Argument

⟨distance⟩: auto — length

Abstand der unteren Linie zur oberen Linien.

Argument

⟨extent⟩: 0pt — length

⟨extent⟩ von underline.

Argument

⟨evade⟩: true — length

⟨evade⟩ von underline.

Argument

⟨body⟩

Zu unterstreichender Text. string | content

```
#squiggly(
  (stroke): 1pt + black,
  (offset): 0pt,
  (amp): 1,
  (period): 1,
  (body)
) → content
```

Zickzack Unterstreichung.

- `#squiggly[#lorem(5)]` → Lorem ipsum dolor sit amet.
- `#squiggly(lorem(5), stroke:2pt+red, offset: 4pt, amp:2, period: 2)` → Lorem ipsum dolor sit amet.

Argument

`(stroke): 1pt + black` — `stroke`

Linienstil für die Unterstreichung.

Argument

`(body)` — `string, content`

Zu unterstreichender Text.

`#highlight((color): yellow, (body)) → content`

Textabschnitt hervorheben (ersetzt highlight).

- `#highlight[#lorem(5)]` → Lorem ipsum dolor sit amet.

Argument

`(color): yellow` — `color`

Farbe der Hervorhebung.

Argument

`(body)` — `string, content`

Hervorzuhebender Inhalt.

`#num((n))`

Deutsches Nummernformat. Basiert auf UNIFY[5].

- `#num(2.3)` → 2,3

`#unit → content`

Formatierung von Größen mit Einheit. Basiert auf UNIFY[6].

[5] https://typst.app/universe/package/unify
[6] https://typst.app/universe/package/unify

- `#unit("m^3")` → m^3

`#si((n), (u))`

SI units mit einem Wert und einer Einheit. Nutzt `#num` und `#unit`.

- `#si(3.5, "m^3")` → 3,5 m^3

Argument

`(n)` `decimal` | `float` | `int`

Wert der Größe.

Argument

`(u)` `str`

Einheit der Größe, wie bei `#unit` beschrieben.

`#eur((n))` → `content`

Formatierung von Eurobeträgen mit dem Euro-Symbol (€).

- `#eur(30.4)` → 30,4 €

Argument

`(n)` `float` | `int` | `decimal`

Geldbetrag

`#degr((n))` → `content`

Formatierung von Gradzahlen.

- `#degr(45)` → 45 °

Argument

`(n)` `float` | `int` | `decimal`

Winkel

`#perc((n))` → `content`

Formatierung von Prozentzahlen.

- `#perc(45)` → 45 %

Argument

`(n)` `float` | `int` | `decimal`

Prozente

II.3.3 Helfer

`#table-fill`

```
#table-fill(
  (fill): white,
  (headerfill): theme.table.header,
  (footerfill): theme.table.header,
  (oddfill): theme.bg.muted,
  (striped): true,
  (headers): 1,
  (footers): 0,
  (colheaders): 0,
  (colfooters): 0,
  (columns): auto,
  (rows): auto,
  (fills): (rows: (:), cols: (:))
)
```

Hilfesfunktion für die Formatierung von Füllfarben für Tabellen.

```
#table(
  columns: 4,
  fill: table-fill(
    footerfill: gradient.linear(..color.map.vlag,
    angle:90deg),
    oddfill: color.map.vlag.first(),
    headers: 2,
    footers: 1,
    colheaders: 1,
    rows: 9
  ),
  ..range(36).map(str)
)
```

0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15
16	17	18	19
20	21	22	23
24	25	26	27
28	29	30	31
32	33	34	35

Argument

⟨fill⟩: white — color | gradient | tiling

Default fill color for cells.

Argument

⟨headerfill⟩: theme.table.header — color | gradient | tiling

Fill color for header rows.

Argument

⟨footerfill⟩: theme.table.header — color | gradient | tiling

Fill color for footer rows.

Argument

⟨oddfill⟩: theta.bg.muted — color | gradient | tiling

Fill color for footer rows.

Part
Anhang
A Dokumentation

Part IV

Index