Etykett

A template for printing onto label sheets with rectangular grids of labels.

v0.1.1 June 25, 2025

https://github.com/SillyFreak/typst-etykett

Clemens Koza

Contents

I Introduction

This template helps you with printing labels, i.e. using sheets of pre-cut adhesive labels that get either the same or individualized content printed onto them.

Here is a simple example for reading data from a CSV file and putting each entry as a label on a sheet:

- Define the content of each label by a function taking a row from the CSV file.
- The shape of the label sheet is defined by the `sheet` parameter of the central `labels()` function, specified via the `sheet()` function.
- like with a regular grid, multiple cells are specified as multiple parameters to the `labels()` function, usually done by "spreading" an array of cells.

```typ
#import "@preview/etykett:0.1.1"

#let data = csv("data.csv").slice(1)

#let name-label((first-name, last-name)) = [
  #set align(center+horizon)
  #set text(14pt)
  Hello, my name is\
  #set text(1.4em)
  *#first-name #last-name*
]

#etykett.labels(
  sheet: etykett.sheet(
    paper: "a4",
    margins: (
      top: 14mm,
      bottom: 15mm,
      x: 6mm,
    ),
    gutters: (x: 2.5mm),
    rows: 9,
    columns: 3,
  ),
…                                                    …
  ..data.map(name-label),
)
```

Check out the parameter documentation of `labels()` and `sheet()` to learn about other capabilities of *Etykett*.

II Module reference

II.a etykett


```
labels(
  sheet: dictionary,
  upside-down: bool,
  inset: length dictionary,
  flipped: bool,
  sublabels: dictionary none,
  border: bool "labels" "sublabels",
  debug: bool,
  ..labels: arguments,
) -> content
```

Renders a grid of labels. This sets up the page, creates the grid, and inserts the label contents.

Parameters:

`sheet` (`dictionary` = `none`) – A label grid sheet definition as created by `sheet()`.

`upside-down` (`bool` = `false`) – Whether to turn the whole grid upside down. This can be useful when there is a need to feed the grid sheets upside-down into the printer.

`inset` (`length` or `dictionary` = `1mm`) – The inset inside the individual labels; useful to make sure the label content fits even if the printer's feeding leads to small alignment inaccuracies. Either a single length or a dictionary of lengths similar to that accepted by `grid()`: valid keys are `left`, `right`, `top`, `bottom`, `x`, `y`, and `rest`. If a dictionary is given, missing sides default to 0pt, but if omitted completely this defaults `1mm`.

`flipped` (`bool` = `false`) – Whether the direction of content in individual (sub)labels is the regular page orientation, or flipped by 90 degrees. This us useful if the label sheet has e.g. wide labels, but they are used as tall labels instead.

`sublabels` (`dictionary` or `none` = `none`) – When set to a dictionary containing integers `rows` and `columns`, each label is further subdivided into a grid as defined. This is useful if the labels on available grid sheets are too big for the desired labels. No insets are added around sublabels inside the same label. If a dictionary is given but a key is missing, that value defaults to 1.

`border` (`bool` or `"labels"` or `"sublabels"` = `false`) – Whether to show the label and/or sublabel bounds by a thin, gray stroke. Possible values are `"labels"` for only showing the labek bounds defined in `sheet()` `"sublabels"` for only showing the sublabel bounds defined by the `sublabels` parameter; or a `bool` for showing both/neither.
Note that this option is intended for debugging or cutting. If your labels should contain a border with a specific stroke, you should include that in the content passed as `labels`.

`debug` (`bool` = `false`) – Deprecated; use the `border` parameter instead.

`..labels` (`arguments`) – The actual label contents to print in each label (or sublabel, if applicable).

```
repeat(n: int, body: content) -> array
```

Inserts a number of identical (sub)labels.

Example:

```typ
#etykett.labels(
  // ... sheet configuration

  ..repeat(3, my-label),
)
```

Parameters:

`n` (`int`) – The number of (sub)labels to insert.

`body` (`content`) – The content of the (sub)labels.

```
skip(n: int) -> array
```

Inserts a number of empty (sub)labels. This is useful for reusing a sheet of labels that has been partially used already.

Example:

```typ
#etykett.labels(
  // ... sheet configuration

  ..skip(3),
  // ... add actual labels
)
```

Parameters:

`n` (`int`) – The number of empty (sub)labels to insert.

```
sheet(
  paper: string dictionary,
  margins: length dictionary,
  flipped: bool,
  gutters: length dictionary,
  rows: int,
  columns: int,
) -> dictionary
```

Returns a dictionary containing the definitions of a label grid sheet.

Parameters:

`paper` (`string` or `dictionary` = `none`) – The paper size, either as a name like `"a4"`, or a dictionary with lengths `width` and `height`.

`margins` (`length` or `dictionary` = `0pt`) – The page margins, either a single length or a dictionary of lengths similar to that accepted by `page()`: valid keys are `left`, `right`, `top`, `bottom`, `x`, `y`, and `rest`. If omitted (or a side in the dictionary is missing), that value defaults to `0pt`.

`flipped` (`bool` = `false`) – Whether the sheet is flipped, i.e. landscape

`gutters` (`length` or `dictionary` = `0pt`) – The gutters between labels, either a single length or a dictionary of lengths with keys `x` and `y`. If omitted (or a value in the dictionary is missing), that value defaults to `0pt`.

`rows` (`int` = `none`) – The number of rows in the grid of labels. The available space is distributed evenly.

`columns` (`int` = `none`) – The number of columns in the grid of labels. The available space is distributed evenly.