Etikett

v0.0.1

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

Clemens Koza

Abstract

Etikett is a template for printing onto label sheets with rectangular grids of labels.

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.

II Module reference

II.a `etikett`


```
labels(
  sheet: dictionary,
  upside-down: bool,
  inset: length dictionary,
  sublabels: dictionary none,
  debug: bool,
  ..labels: arguments,
)
```

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`.

`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.

`debug` (`bool` = `false`) – Whether to show the label and sublabel bounds by a thin, gray stroke.

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

```
repeat(n: int, body: content)
```

Inserts a number of identical (sub-)labels.

Parameters:

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

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

```
skip(n: int)
```

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

Parameters:

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

```
sheet(
  paper: string dictionary,
  margins: length dictionary,
  gutters: length dictionary,
  rows: int,
  columns: int,
)
```

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` = `none`) – 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`.

`gutters` (`length` or `dictionary` = `none`) – 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.