The `leporello` class

A simple LaTeX document class to create folded leaflets using columns and boxes

Jasper Habicht [*]

Version 0.8.0, released on 12 August 2025

---

1 Introduction

The `leporello` class is a simple LaTeX document class to create folded leaflets with the following key features:

- The document layout consists of a specific number of pages that can have varying widths and are placed next to each other. Each page contains one frame to contain typeset material.
- Material is typeset in boxes and boxes are positioned in columns which in turn are placed into the frames on the pages of the document.
- Columns are predefined and then placed into the frame of a page in the document layout. This way, columns can be used on pages with varying widths.
- Boxes can have padding (including bleed) and a background and are positioned inside a column.

A folded leaflet is sometimes called leporello in reference to the servant of Don Giovanni (known from the opera by Mozart) who, at one point, unfolds a lengthy zig-zag folded list of his master's love affairs.

[*] E-mail: mail@jasperhabicht.de. The title image was generated using ChatGPT.

2 Loading the document class

Use `\documentclass{leporello}` to load the document class. The document class loads the `geometry`, the `graphicx` and the `l3draw` package.

`\leporelloset`{‹options›}

Use `\leporelloset` or use the optional argument of the `\documentclass` command to globally set the document layout. The following options are available:

`columns`={‹list of dimensions›}
`two columns`
`three columns`
`four columns`
`four columns wrap`
`five columns`
`six columns`

Expects a comma-separated list of dimensions that describe the widths of the columns in the layout. If `columns` is not specified, the class assumes the default value of `97mm, 100mm, 100mm`. The width of the document layout is automatically calculated from the dimensions set via `columns`.

A few column layouts are predefined. These do not expect a value. The layouts will set the following list of dimensions:

Key	List of dimensions
`two columns`	99 mm, 99 mm
`three columns`	97 mm, 100 mm, 100 mm
`four columns`	99 mm, 99 mm, 99 mm, 100 mm
`four columns wrap`	97 mm, 99 mm, 100 mm, 101 mm
`five columns`	97 mm, 97 mm, 97 mm, 99 mm, 100 mm
`six columns`	99 mm, 99 mm, 96 mm, 96 mm, 100 mm, 100 mm

`layout height`={‹dimension›}

Expects a dimension that sets the height of the final layout. If `layout height` is not specified, the class assumes the default value of `210mm` (the height of A4 landscape).

`show frames`

If set, frames around the single pages (columns) are shown. Does not expect a value.

`show ids`

If set, the IDs of the boxes are shown. Does not expect a value.

`prepress`

If set, the paper size is increased at all four pages by the width of the info area and crop marks are shown. Does not expect a value.

`info area`={‹dimension›}

Expects a dimension for the width of the info area that is shown if `prepress` is set. If `info area` is not specified, the class assumes the default value of `10mm`.

```
bleed={‹dimension›}
```

Expects a dimension for the width of the bleed that is added to boxes with background and other material reaching to the edges of the document layout. If `bleed` is not specified, the class assumes the default value of `3mm`.

```
\leporellobleed
```

Retrieves the width of the bleed in pt.

```
\leporelloboxwidth
```

Retrieves the width of the current box including padding, but without bleed, in pt.

3 Defining columns

Columns are grouped material to be typeset onto a specific page of the folded leaflet. A column can contain anything that can be placed in a TeX box in general, but the typical use of a column is to only serve as a frame to which boxes are attached. While the package provides a flow mechanism across boxes, no such mechanism is provided for columns. Thus, if a column contains more than fits into it, the relevant parts just spill over the frame edges.

```
\begin{leporellocolumn}{‹string›}
  ‹body›
\end{leporellocolumn}
```

A column is defined by the `leporellocolumn` environment which takes one mandatory argument taking the name (ID) of the column.

4 Positioning boxes

```
\begin{leporellobox}[‹options›]
  ‹body›
\end{leporellobox}
```

Boxes can be positioned inside of columns. They can contain anything that can be placed in a TeX box in general. A box is defined by the `leporellobox` environment which takes one optional argument to set box-specific options. These are the following:

```
align parent={‹tuple of poles›}
```

Expects a comma-separated list of two items (a tuple) which denotes the horizontal and vertical pole of which the intersection defines the coordinate of the current column that serves as anchor to align the current box. If not specified, the default value `l, t` is assumed, denoting the top left corner. Available poles are:

Key	Meaning

`l`	left edge of the box
`hc`	horizontal center of the box
`r`	right edge of the box
`t`	top edge of the box
`vc`	vertical center of the box
`B`	baseline of the box
`b`	bottom edge of the box

align self={‹tuple of poles›}

Expects a comma-separated list of two items (a tuple) which denotes the horizontal and vertical pole of which the intersection defines the coordinate of the current box that serves as anchor to align the current box to the current column. If not specified, the default value `l, t` is assumed, denoting the top left corner. Available poles are the same as for `align parent`.

offset={‹tuple of dimensions›}

Expects a comma-separated list of two dimensions (a tuple) which defines the offset of the anchor set via `align parent` and `align self`. The first dimension is the offset to the right, the second dimension the offset downwards. If not specified, the default value `0mm, 0mm` is assumed.

width={‹dimension›}

Expects a dimension to explicitly set the width of the current box. If not specified, the box is as wide as the parent column.

height={‹dimension›}

Expects a dimension to explicitly set the height of the current box. If not specified, the box takes its natural height which means that it is as high as necessary to fit the contents.

stretch height

If set, the box is stretched until its relevant edge (the bottom edge if aligned at the top, the top edge if aligned at the bottom) meets the edge of the parent column. Does not expect a value.

padding left={‹dimension›}
padding right={‹dimension›}
padding top={‹dimension›}
padding bottom={‹dimension›}
padding={‹key-value list›}
no padding

`padding left`, `padding right`, `padding top` and `padding bottom` each expect a dimension to describe the padding of the contents from the relevant edge of the box. If not specified, the default value of `7.5mm` is assumed.

All four padding settings can also be stated using the `padding` key and the subkeys `left`, `right`, `top`, `bottom`. Using this syntax, the default value of the padding setting would be expressed as `padding={left=7.5mm, right=7.5mm, top=7.5mm, bottom=7.5mm}`.

The key `no padding` sets all paddings to zero. This key does not expect a value.

pre={‹code›}

Expects a token list that is placed before the actual contents of the box. Should not contain typeset material. This key should be used if boxes are manually split using the `\leporelloboxbreak` command.

background color={‹color name›}
background color={none}

Expects a color name as defined via `l3color` or `none` which will not fill the background. If not specified, the default value of `none` is assumed.

background code={‹code›}

Expects typeset material that will be placed into the background of the box aligned at the upper left corner of the box. THe typeset material is clipped to the size of the box.

bleed={‹list of values›}

Expects a comma-separated list consisting of up to four items with the values `l` and `r`, `t` and `b` that describe the edges (left, right, top and bottom) where bleed should be added to the box. Note that bleed is never added to the inner edges where the columns meet.

store width={‹control sequence›}

Expects a single control sequence (macro) to store the width of the current box.

store height={‹control sequence›}

Expects a single control sequence (macro) to store the height of the current box.

flow into={‹integer›}

Expects the ID of the box into which typeset material will flow into if it does not fit into the current box. See section 6 below.

5 Typesetting boxes into columns

\leporellotypesetcolumns[‹options›]{‹list of strings›}

Using the command `\leporellotypesetcolumns` which takes one mandatory argument, previously defined columns can be placed onto a page of the document. The command expects as argument a comma-separated list of names of previously defined columns. These are then placed onto a document page from left to right while the width is taken from the setting via the `columns` key and the height is taken from the setting via the `layout height` key.

reverse layout

The command takes one optional argument that can take the option `reverse layout`. If set, the widths of the columns are reversed, but the placement of the columns is still from left to right. This

option does not take a value. The option should be used to typeset the verso of a folded leaflet that naturally has the widths of the columns reversed.

Currently, boxes can only be attached to columns. A future version of the document class may allow the attachment of boxes to each other. If a box should be positioned below another box with natural height defined by its contents, the `store height` key can be used to store the height of the upper box into a custom macro which can then be used to calculate the offset from the top.

6 Flow mechanism

Typeset material can flow from one box to another box. To this end, the key `flow into` can be set to the relevant box and assigned the ID of the box to flow into as value. The IDs of the boxes are integers starting from 1 and increasing in the order of the definition of the boxes. By setting the key `show ids` to the document options, the IDs of all boxes are shown.

`\leporelloboxbreak`

The flow mechanism works across multiple boxes. But due to the way the typesetting mechanism of TeX works, it needs some manual adjustment if the typeset material flows across boxes of different width. In this case, the command `\leporelloboxbreak` should be inserted at the point where the break should take place.

6.1 Hooks

The package offers two pairs of hooks that are positioned at the start and end of columns and boxes allowing for inserting code. The package uses the hook `leporello/box/before` to insert code for the flow mechanism.

Apart from that, the package uses the hook `begindocument/before` to insert layout-related settings among other things via the `geometry` package as well as the hook `shipout/foreground` to insert code to draw cropmarks.

Hook	Position
`leporello/column/before`	before the code block stored via the `leporellocolumn` environment is typeset to the column box, preceded by a line of code that sets `\l_leporello_current_column_str` to the name of the current column
`leporello/column/after`	after the code block stored via the `leporellocolumn` environment is typeset to the column box
`leporello/box/before`	before the code block stored via the `leporellobox` environment is typeset to the relevant box which is preceded by a line of code that sets `\l_leporello_current_box_int` to the ID of the current column and inserts the code defined via the `pre` key
`leporello/box/after`	after the code block stored via the `leporellobox` environment is typeset to the relevant box

7 Other settings

7.1 Inserting images

One way to insert images is via the `\includegraphics` command provided by the `graphicx` package. Using the `background code` key, images can be added to the background of a box.

```
\leporelloimage[‹options›]{‹file name›}
```

With the command `\leporelloimage` images that cover full boxes can be inserted. This command can be placed in a `leporellobox` with zero padding and it should only be used inside a `leporellobox` environment. The command has one mandatory argument that takes a relative path and file name to select the image to be inserted. It also has one optional argument to take the following options:

```
clip width={‹dimension›}
clip height={‹dimension›}
```

Expect a dimension depicting the width and the height of the boxed image.

```
scale={‹floating point number›}
```

Expects a floating point number depicting the scaling factor of the image. This factor will not affect the size of the box.

```
width={‹dimension›}
height={‹dimension›}
```

Expect a dimension to explicitly set the width and the height of the image. Setting `width` will override a `scale` value. Setting `height` will override a `width` or `scale` value. The aspect ratio of the original image will always be kept.

```
offset={‹tuple of dimensions›}
```

Expects a comma-separated list consisting of two values (tuple) that describe the offset of the image that will be positioned per default so that the upper left corner sits at the upper left corner of the box. A positive offset will shift the image in upper left direction.

```
ignore padding={‹list of values›}
```

Expects a comma-separated list consisting of up to four items with the values `l` and `r`, `t` and `b` that describe the edges (left, right, top and bottom) where the image should ignore the padding of the current box.

```
fill bleed
```

If set, the image will spread into the bleed. This will result in a shift of the image by the size of the bleed which may need to be accounted for using `offset`. Does not expect a value.

7.2 Defining colors

The package uses the color model of the `l3color` module. The `xcolor` package is not supported. To provide a user interface to define and select colors, the commands `\leporellocolordefine` and `\leporellocolorselect` are defined.

```
\leporellocolordefine{‹string›}{‹color model›}{‹list of values›}
```

The command `\leporellocolordefine` takes three arguments, the first being the name of the color to be defined. The second argument takes the color model (for example `rgb` or `cmyk`) and the third argument takes the color values. For more information about which color models are supported, please refer to the documentation of the `l3color` module.

`\leporellocolorselect`{‹string›}

The command `\leporellocolorselect` takes the name of the previously defined color as argument. All following objects are affected by this color setting. To colorize only a few letters, use curly braces for grouping.

7.3 Defining styles

`\leporellosetstyle`{‹string›}{‹key-value list›}

In order to simplify the setting of recurring options to the `leporellobox` environment, it is possible to group several of these options as style via the `\leporellosetstyle` command which takes as first argument the name of the newly defined style and as second argument the releevant options (key-value pairs). The style can then be used like an option to any `leporellobox` environment.

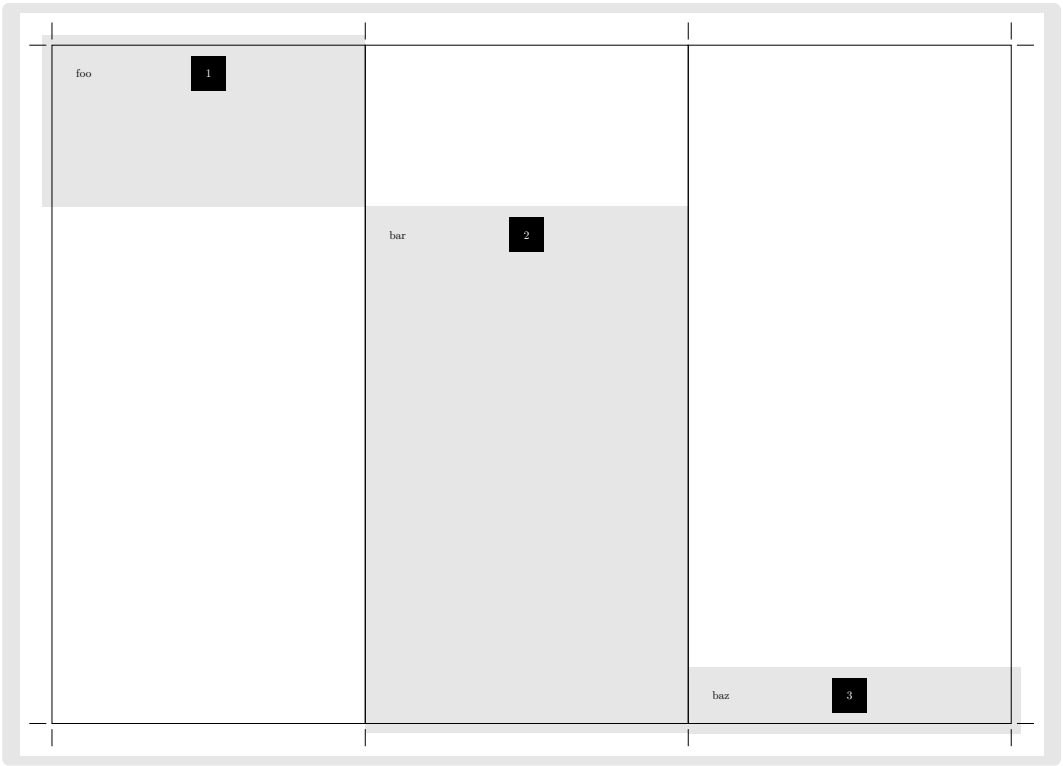
7.4 Restoring justified typesetting

`\leporellojustified`

In order to restore the default justified typesetting style of TeX after having set `\raggedright`, `\raggedleft` or `\centering`, the packages defines the command `\leporellojustified`.

8 Example

The following example shows some of the basic ideas of the package by providing a code example and showing its output.


foo
1
bar
2
baz
3

```
\documentclass[
  prepress,
  show frames,
  show ids
]{leporello}

\leporellosetstyle{align bottom}{
  align parent={l,b},
  align self={l,b}
}

\begin{leporellocolumn}{example-a}
\begin{leporellobox}[
  background color=black!10,
  height=50mm,
  bleed={l,t}
]
foo
\end{leporellobox}
\end{leporellocolumn}

\begin{leporellocolumn}{example-b}
\begin{leporellobox}[
  background color=black!10,
  offset={0mm,50mm},
  stretch height,
  bleed={b}
]
bar
\end{leporellobox}
\end{leporellocolumn}

\begin{leporellocolumn}{example-c}
\begin{leporellobox}[
  background color=black!10,
  align bottom,
  bleed={r,b}
]
baz
\end{leporellobox}
\end{leporellocolumn}

\begin{document}

\leporellotypesetcolumns{
  example-a,
  example-b,
  example-c
}

\end{document}
```

The example shows the use of the `prepress` key that adds cropmarks to the layout. It also shows how the `show frames` key adds frames around the columns. Finally, the `show box` key adds the box IDs to the boxes.

The default layout has three pages which are shown here. Three columns named `example-a`,

`example-b` and `example-c` are defined and contain one box each. A custom style is defined to combine the two keys needed to bottom-align the third box. A fixed height is set to for the first box and the second box has and offset and its height stretched to the bottom of the column. Bleed is added to the relevant edges of the three columns.

9 Changes

v0.7.0 (2025/08/08) First public beta release.

v0.8.0 (2025/08/12) Added option to place arbitrary code to box background.