Columns - Multiple columns support in Pandoc's markdown

Julien Dutant

Columns

Multiple columns support in Pandoc's markdown.

Introduction

This Lua filter for Pandoc provides a flexible markdown syntax for multicolumn support in Pandoc targetting both HTML and LaTeX/PDF output. Features:

- Multiple markdown syntaxes ("three-columns" Div, nested "columns" and "column" Div, "columns" with explicit column breaks)
- Column breaks can be automatic or explicit
- Spanning elements breaking across all columns
- Customizing gaps and separators
- Automatically provides CSS header / LaTeX preamble
- Automatic typographic adjustements (avoid empty space at the top of the first column which sometimes appears in HTML).
- Recursive (multi-columns within multi-columns)

Html output relies on CSS Multi-column layout and LaTeX/PDF outputs on the `multicol` LaTeX package.

Limitations: in `html` output, support is limited to recent browsers and variable across browsers.

This document also serves as a test document. To see the multi-columns layouts of this document in action, you need to process it with `pandoc` using this filter.

NOTE This README.md is a demonstration file, it is better viewed as PDF.

Pre-requistes

Requires Pandoc. Copy the file `columns.lua` in your working folder or in Pandoc's `filter` folder. Called from the command line with a `-L` or `--lua-filter`

option:

```
pandoc --lua-filter columns.lua SOURCE.md -o DESTINATION.html
```

```
pandoc -L columns.lua SOURCE.md -o DESTINATION.pdf
```

Or from a `filters` field in a Pandoc defaults file. See the Pandoc documentation for further details.

For instance, to process the present documentation use:

```
pandoc -L columns.lua README.md -o readme.html
```

```
pandoc -L columns.lua README.md -o readme.pdf
```

Basic usage

Columns

In Pandoc markdown source specify a multicolumn section as follows:

```
::: columns

...content that will be spread over several columns...

:::
```

The filter will render this section as a multicolumns layout in `html` and LaTeX, as illustrated below (you need to process this document with `pandoc` using this filter to see the results in `html` or `pdf`:

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Donec a ante in mi ornare volutpat sed sit amet diam. Nullam interdum erat a augue faucibus, nec tempus tortor sagittis. Aenean imperdiet imperdiet dignissim. Nam aliquam blandit ex, sed molestie nibh feugiat ac. Morbi feugiat convallis semper. Ut et consequat purus. Fusce convallis vehicula enim in vulputate. Curabitur a augue arcu. Mauris laoreet lectus arcu, sed elementum turpis scelerisque id. Etiam porta turpis quis ipsum dictum vulputate. In ut convallis urna, at imperdiet nunc. Cras laoreet, massa lobortis gravida egestas, lacus est pellentesque arcu, imperdiet efficitur nibh dolor vel sapien. Sed accumsan condimentum diam non pellentesque.

Vestibulum cursus nisi risus, sit amet consectetur massa suscipit nec. Sed condimentum, est id iaculis ornare, purus risus finibus felis, posuere congue est nibh eget dui. Maecenas orci erat, commodo auctor justo quis, vestibulum mollis ex. Vivamus sed bibendum turpis. Donec auctor, leo a cursus efficitur, quam urna dignissim enim, viverra condimentum orci est non sem. Donec ac viverra nisl. Suspendisse ac auctor massa. Mauris porttitor purus vel velit vehicula, sed efficitur odio lacinia. Fusce sed odio arcu. Ut rhoncus lacus vel magna interdum tincidunt. Nunc imperdiet finibus tincidunt.

This syntax is based on the `fenced_div` syntax of Pandoc's' markdown. At least three consecutive colons are needed, both at the beginning and at then end of your multi-column section (even if it runs until the end of your document). But more than three are fine:

```
::: columns ::::::

...content that will be spread over several columns...

::::::::::::::
```

Each opening series of colons needs to be matched with a closing ones. For readibility we usually match their number of colons but it's not necessary (as the above illustrates). If you enclose sections within sections (see container syntax, nesting, column spans and column breaks below) you need to make sure that each opening series of colons is matched by a closing one, otherwise Pandoc will not recognize them or interpret them incorrectly.

Here `columns` is a *attribute* of the fenced div (section). As we'll see below, these sections can have more than a single attribute. When they have several, they need to be specified within curly brackets and `columns` should be preceded by a dot, as in:

```
::::: {.columns .someattribute property=value}

...content that will be spread over several columns...

:::::
```

Beware of Divs in fluid columns

With fluid columns, i.e. no explicit line breaks, browsers decide where to put line breaks. Beware though that Divs elements within a column are counted as unbreakable blocks in most browsers. For instance, the following places a Div with classes ".only-in-format .html" within a fluid multiple columns:

```
::::: columns

::: {.only-in-format .html}

First paragraph (...)

Second paragraph (...)

Third paragraph (...)

:::
```

```
:::::
```

You might expect the columns to break between one of these paragraphs or within them. But they won't: browsers will usually treat the entire three-paragraph Div as one block that will stay in a single column. Solutions: either move the contained outside, or break it into multiple ones.

Moving it outside:

```
::: {.only-in-format .html}

::::: columns

First paragraph (...)

Second paragraph (...)

Third paragraph (...)

:::::

:::
```

Breaking it up:

```
::::: columns

::: {.only-in-format .html}

First paragraph (...)

:::

::: {.only-in-format .html}

Second paragraph (...)

:::

::: {.only-in-format .html}

Third paragraph (...)

:::

:::::
```

Specifying the number of columns

By default two columns are provided. You can specify the desired number of columns in various ways:

```
::: twocolumns

::: three-columns

::: five_columns

::: {.columns column-count=3}
```

Note that in `html` browsers may override your specified number of columns.

Ragged columns (LaTeX output only)

Default LaTeX/PDF output justifies columns vertically. That is, if columns are explicitly broken at certain points, LaTeX ensures that the text in each column occupies its full height by stretching inter-paragraph space. In HTML output columns are always "ragged", that is, inter-paragraph space isn't stretched and shorter columns have blank space at the end.

If you want ragged columns in LaTeX, you can set this globally in the document's metadata or on locally on a give `columns` Div. In the document data, either of these keys will work:

```
ragged-columns: true
raggedcolumns: true
```

Locally, add the `ragged` (or `raggedcolumns` or `ragged-columns`) class to a `columns` Div:

```
::::: {.columns .ragged}

...

:::::
```

Note that this doesn't work on individual `column` Divs, only on the `columns` Div that contains them.

There is a corresponding `justifiedcolumns` (alias `justified-columns`) global setting and a `justified` (alias `justifiedcolumns`, `justified-columns`) class for specific `columns` Div.

This column is vertically short.

This column is vertically short.

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Donec a ante in mi ornare volutpat sed

sit amet diam. Nullam interdum erat a augue faucibus, nec tempus tortor sagittis. Aenean imperdiet imperdiet dignissim. Nam aliquam blandit ex, sed molestie nibh feugiat ac. Morbi feugiat convallis semper. Ut et consequat purus. Fusce convallis vehicula enim in vulputate.

Now in ragged columns mode:

This column is vertically short.

This column is vertically short.

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Donec a ante in mi ornare volutpat sed sit amet diam. Nullam interdum erat a augue faucibus, nec tempus tortor sagittis. Aenean imperdiet imperdiet dignissim. Nam aliquam blandit ex, sed molestie nibh feugiat ac. Morbi feugiat convallis semper. Ut et consequat purus. Fusce convallis vehicula enim in vulputate.

Customizing the gap and rule between columns

The gap and rule between columns can be customized too. The gap is specified with a `columngap` (or `column-gap` or `columnsep` or `column-sep`) attribute. The rule is specified with a `column-rule` (or `columnrule`) attribute using CSS syntax.

```
::: {.columns columngap=3em column-rule="1px solid black"}
```

```
::: {.threecolumns columngap=4em column-rule="3pt solid blue"}
```

Here is an illustration:

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Donec a ante in mi ornare volutpat sed sit amet diam. Nullam interdum erat a augue faucibus, nec tempus tortor sagittis. Aenean imperdiet imperdiet dignissim. Nam aliquam blandit ex, sed molestie nibh feugiat ac. Morbi feugiat convallis semper. Ut et consequat purus. Fusce convallis vehicula enim in vulputate. Curabitur a augue arcu. Mauris laoreet lectus arcu, sed elementum turpis scelerisque id. Etiam porta turpis quis ipsum dictum vulputate. In ut convallis urna, at imperdiet nunc. Cras laoreet, massa

lobortis gravida egestas, lacus est pellentesque arcu, imperdiet efficitur nibh dolor vel sapien. Sed accumsan condimentum diam non pellentesque.

Vestibulum cursus nisi risus, sit amet consectetur massa suscipit nec. Sed condimentum, est id iaculis ornare, purus risus finibus felis, posuere congue est nibh eget dui. Maecenas orci erat, commodo auctor justo quis, vestibulum mollis ex. Vivamus sed bibendum turpis. Donec auctor, leo a cursus efficitur, quam urna dignissim enim, viverra condimentum orci est non sem. Donec ac viverra nisl. Suspendisse ac auctor massa. Mauris porttitor purus vel velit vehicula, sed efficitur odio lacinia. Fusce sed odio arcu. Ut rhoncus lacus vel magna interdum tincidunt. Nunc imperdiet finibus tincidunt.

Spanning elements

Elements that span across all columns are introduced as `column-span` (or `columnspan`) sections:

```
::: columns ::::::::

content in columns

::::: column-span

# This heading spans across all columns

:::::

content in columns

:::
```

Here is an illustration:

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Donec a ante in mi ornare volutpat sed sit amet diam. Nullam interdum erat a augue faucibus, nec tempus tortor sagittis. Aenean imperdiet imperdiet dignissim. Nam aliquam blandit ex, sed molestie nibh feugiat ac. Morbi feugiat convallis semper. Ut et consequat purus. Fusce convallis vehicula enim in vulputate. Curabitur a augue arcu. Mauris laoreet lectus arcu, sed elementum turpis scelerisque id. Etiam porta turpis quis ipsum dictum vulputate. In ut convallis urna, at imperdiet nunc. Cras laoreet, massa lobortis gravida egestas, lacus est pellentesque arcu, imperdiet efficitur nibh dolor vel sapien. Sed accumsan condimentum diam non pellentesque.

Vestibulum cursus nisi risus, sit amet consectetur massa suscipit nec

Sed condimentum, est id iaculis ornare, purus risus finibus felis, posuere congue est nibh eget dui. Maecenas orci erat, commodo auctor justo quis, vestibulum mollis ex. Vivamus sed bibendum turpis. Donec auctor, leo a cursus efficitur, quam urna dignissim enim, viverra condimentum orci est non sem. Donec ac viverra nisl. Suspendisse ac auctor massa. Mauris porttitor purus vel velit vehicula, sed efficitur odio lacinia. Fusce sed odio arcu. Ut rhoncus lacus vel magna interdum tincidunt. Nunc imperdiet finibus tincidunt.

Explicitly specifying column breaks

Column breaks can be explicitly specified. This can be done using `\columnbreak` or a `columnbreak` (or `column-break`) section.

```
::: columns

This content is in a first column.

\columnbreak

This content is in a second column.

:::: columnbreak
::::

This content is in a third column.

:::: column-break
::::

This content is in a fourth column.

:::
```

The result is:

This content is in a first column.

This content is in a second column.

This content is in a third column.

This content is in a fourth column.

Warning and limitations

- In `html`, browsers may ignore explicit column breaks.
- A `\columnbreak` break must be preceded by an empty line and occupy a line on its own.
- A `::: columnbreak` break must be followed by a closing line of `:::`.

When columnbreaks are explicitly specified, they are used to determine the number of columns. If the section both speficies a number of columns and includes explicit columnbreaks, the greatest number is used.

Container syntax

A multicolumn section with explicit breaks can also be written using a container syntax, with `column` sections included in a `columns` section, as follows.

```
:::::: columns

::: column

First column content here

:::

::: column

Second column content

:::

:::::
```

This follows Pandoc's markdown syntax for `beamer` output. Note that individual column widths and further column attributes available in `beamer` outputs are not supported here.

Container syntax and columnbreak syntax can be mixed, as in the example below:

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Donec a ante in mi ornare volutpat sed sit amet diam. Nullam interdum erat a augue faucibus, nec tempus tortor sagittis. Aenean imperdiet imperdiet dignissim. Nam aliquam blandit ex, sed molestie nibh feugiat ac. Morbi feugiat convallis semper. Ut et consequat purus. Fusce convallis vehicula enim in vulputate. Curabitur a augue arcu.

Mauris laoreet lectus arcu, sed elementum turpis scelerisque id. Etiam porta turpis quis ipsum dictum vulputate. In ut convallis urna, at imperdiet nunc. Cras laoreet, massa lobortis gravida egestas, lacus est pellentesque arcu, imperdiet efficitur nibh dolor vel sapien. Sed accumsan condimentum diam non pellentesque.

Vestibulum cursus nisi risus, sit amet consectetur massa suscipit nec. Sed condimentum, est id iaculis ornare, purus risus finibus felis, posuere congue est nibh eget dui. Maecenas orci erat, commodo auctor justo quis, vestibulum mollis ex.

Advanced usage

Nesting

Multicolumn sections can be nested. Support for nesting may vary across browsers. Here is an illustration:

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Donec a ante in mi ornare volutpat sed sit amet diam. Nullam interdum erat a augue faucibus, nec tempus tortor sagittis. Aenean imperdiet imperdiet dignissim. Nam aliquam blandit ex, sed molestie nibh feugiat ac. Morbi feugiat convallis semper. Ut et consequat purus. Fusce convallis vehicula enim in vulputate. Curabitur a augue arcu.

Mauris laoreet lectus arcu, sed elementum turpis scelerisque id. Etiam porta turpis quis ipsum dictum vulputate. In ut convallis urna, at imperdiet nunc.

This is a two-column section nested within the middle column of a three-column section.

Cras laoreet, massa lobortis gravida egestas, lacus est pellentesque arcu, imperdiet efficitur nibh dolor vel sapien. Sed accumsan condimentum diam non pellentesque.

Vestibulum cursus nisi risus, sit amet consectetur massa suscipit nec. Sed condimentum, est id iaculis ornare, purus risus finibus felis, posuere congue est nibh eget dui. Maecenas orci erat, commodo auctor justo quis, vestibulum mollis ex.

Number of columns

Number of columns can be specified in English up to ten. Accepted patterns are `<number>columns`, `<number>-columns` and `<number>_columns`. Note that this is a "class", and should be preceded by a dot when specified along other attributes within curly brackets:

```
::: twocolumns

::: {.three-columns columnsep=2em}

:::
```

Alternatively, the `column-count` can be used to specify any number of columns.

```
::: {.columns column-count=3}
```

If both English names and `column-count` are used, the former prevails.

HTML output

The `html` output looks like this. Without column breaks:

```
<div class="columns" style="column-count: 2; column-rule: 1px solid black;">

Content that distributed in columns...

<div class="column-span" style=";">
Content that spreads across all columns
</div>

More content distributed in columns...

</div>
```

With columnbreaks:

```
<div class="columns" style="column-count: 2;">

Content of the first column.

<div style="break-after: column;"></div>

Content of the second column.

</div>
```

In CSS `break-after: column` means "after this element, place a column break".

The classes `columns` and `column-span` are needed to ensure that the first element of a multiple columns `div`, or the first element after an element spanning across columns, have no top margin. If they had we would get unwanted space at the beginning of the first column. Thus the filter adds the following to the header:

```
<style>
  .columns :first-child {margin-top: 0;}
  .column-span + * {margin-top: 0;}
</style>
```

LaTeX output

The LaTeX output looks as follows. Preamble:

```
\usepackage{multicol}
```

Document body:

```
{\begin{multicols}{2}

content distributed over two columns
```

```
\end{multicols}
}
```

With properties and explicit column breaks:

```
{\setlength{\columnsep}{4em}
\setlength{\columnseprule}{ 3pt}
\renewcommand{\columnseprulecolor}{\color{blue}}
\begin{multicols}{3}

content distributed over three columns

\end{multicols}
}
```

Note that the `multicols` environment is wrapped within `{...}`. This is to ensure that settings of `\columnsep`, `\columnseprule` and `\columnseprulecolor` do not affect subsequent `multicol` environments.

Contributing

Issues and pull requests are welcome. They can be submitted to the repository.

Related

- pandoc-columns, a Pandoc filter written in Haskell.

References

- `html`: CSS Multi-column layout
- LaTeX: `multicol` LaTeX package
- Pandoc: https://pandoc.org/lua-filters.html
- Pandoc lua filters: https://pandoc.org/lua-filters.html