

Table of Contents

1. Tables	2
1.1 Enable the extension	2
1.2 Set column widths	2
1.2.1 Percentages that add up to 100%	2
1.2.2 Percentages that don't add up to 100%	3
1.2.3 Fixed widths for every column	4
1.2.4 Mixing percentages and fixed widths	4
1.2.5 Left-aligning a header	5
1.3 Configure table layouts	6
1.3.1 Use a compact layout	6
1.3.2 Use more than one header row	7
1.3.3 Merge cells	8
1.3.4 Fix a table that renders as pipe characters	9
1.3.5 Adjust cell shading	9
1.3.6 Align content within a tall row	10
1.3.7 Rotate headings	10
1.4 Reference	11
1.4.1 Syntax	11
1.5 Customise with a CSS style sheet	12

1. Tables

`prodockit.tables` adds layout controls to an ordinary Markdown table. You can change column widths, reduce spacing, use more than one header row, merge cells, align content within tall rows, and rotate long headings.

1.1 Enable the extension

Enable it in `zensical.toml`:

```
[project.markdown_extensions."prodockit.tables"]
```

Choose the feature that solves the table's problem:

[Table 24.1](#) maps common table-layout problems to the attribute that solves each one.

Table 1.1. Enable the extension

Need	Attribute
Set a column or grouped-header width	<code>width="30%"</code> or a fixed width such as <code>8rem</code>
Fit many short columns	<code>.compact</code>
Repeat more than one header row	<code>.header</code>
Merge cells	<code>colspan=2</code> or <code>rowspan=2</code>
Change one cell's shading	<code>shade="off"</code> or <code>shade="8%"</code>
Align one cell vertically	<code>valign="top"</code> , <code>valign="middle"</code> or <code>valign="bottom"</code>
Turn a long heading vertically	<code>rotate=90</code> or <code>rotate=270</code> , with <code>width</code>

The next examples show each feature in isolation before combining them.

1.2 Set column widths

Choose percentage widths when columns should share the available page and fixed widths when an element must retain a physical size. The examples progress from one sizing system to combinations of both.

1.2.1 Percentages that add up to 100%

Give every column an explicit percentage and they're used exactly as written:

Markdown

```
| Name {: width="25%" } | Description {: width="50%" } | Due {: width="25%" } |  
| ---|---|---|  
| Headings | Heading ids and section numbers | Q1 |  
| Refs | Cross-references, resolved by number | Q2 |
```

Result

The rendered table in [Table 24.2](#) shows three columns using their exact percentage widths.

Table 1.2. Percentages that add up to 100%

Name	Description	Due
Headings	Heading ids and section numbers	Q1
Refs	Cross-references, resolved by number	Q2

1.2.2 Percentages that don't add up to 100%

Leave a column without a width and it uses the remaining space. If several columns have no width, they share that space evenly:

Markdown

```
| Name {: width="20%" } | Description | Due {: width="15%" } |  
| ---|---|---|  
| Headings | Heading ids and section numbers | Q1 |  
| Refs | Cross-references, resolved by number | Q2 |
```

Result

The rendered table in [Table 24.3](#) shows the browser preserving the requested proportions when the percentages do not total 100%.

Table 1.3. Percentages that don't add up to 100%

Name	Description	Due
Headings	Heading ids and section numbers	Q1
Refs	Cross-references, resolved by number	Q2

`Name` and `Due` get the widths given; `Description`, left unannotated, takes the remaining 65%. A column left unannotated in a table with no `width` anywhere at all is completely untouched, though - only a table with at least one `width` gets a `<colgroup>`.

1.2.3 Fixed widths for every column

A fixed width is useful when a column should stay the same size even when the page becomes wider or narrower. You can give every column a fixed width:

Markdown

```
| Icon {: width="60px" } | Description {: width="200px" } | Format {:
width="100px" } |
| ---|---|---|
| :material-file-pdf-box: | A downloadable PDF | PDF |
| :material-file-document: | A Markdown source file | Markdown |
```

Result

The rendered table in [Table 24.4](#) shows fixed-width columns alongside their content.

Table 1.4. Fixed widths for every column

Icon	Description	Format
	A downloadable PDF	PDF
	A Markdown source file	Markdown

1.2.4 Mixing percentages and fixed widths

Percentage and fixed-width columns can appear in the same table. A column can still be left without a width and use the remaining space:

Markdown

```
| # {: width="40px" } | Name {: width="50%" } | Description |
|---|---|---|
| 1 | prodockit.headings | Heading ids and section numbers |
| 2 | prodockit.tables | Column widths on a table |
```

Result

The rendered table in [Table 24.5](#) shows percentage and fixed widths used together.

Table 1.5. Mixing percentages and fixed widths

#	Name	Description
1	prodockit.headings	Heading ids and section numbers
2	prodockit.tables	Column widths on a table

1.2.5 Left-aligning a header

By default a header cell is centred and a body cell is left-aligned - the browser's own default styling for `<th>` / `<td>`, unrelated to `prodockit.tables`. To left-align a header too, use Python-Markdown's own column-alignment syntax - a `:` on the left side of that column's own dashes in the separator row - which applies to the header *and* every body cell in that column alike, and combines with `width` on the same header cell with no conflict:

Markdown

```
| Name {: width="30%" } | Description |
|:---|---|
| Headings | Heading ids and section numbers |
| Refs | Cross-references, resolved by number |
```

Result

The rendered table in [Table 24.6](#) shows a selected heading aligned differently from the others.

Table 1.6. Left-aligning a header

Name	Description
Headings	Heading ids and section numbers
Refs	Cross-references, resolved by number

`:---: / ---:` center- or right-align a column the same way - see [Python-Markdown's own tables docs](#) for the full syntax. This isn't a `prodockit.tables` feature; it's documented here because it's the natural companion to `width` when sizing a column, not something `prodockit.tables` needs to reimplement.

1.3 Configure table layouts

There are no additional `prodockit.tables` settings in `zensical.toml`. Configure each table in its Markdown by adding the attributes shown in the examples below.

1.3.1 Use a compact layout

A table with many short columns can become wider than the page. Add `.compact` to reduce the minimum column width and cell spacing.

Mark it `{: .compact }` on any header cell:

Markdown

```
| Threat {: .compact } | Likelihood | Impact | Risk |
|---|---|---|---|
| Credential theft | H | H | H |
```

Result

The rendered table in [Table 24.7](#) shows the reduced spacing produced by the compact layout.

Table 1.7. Use a compact layout

Threat	Likelihood	Impact	Risk
Credential theft	H	H	H

Use it only when the normal table is too wide. Put the marker on any header cell. It affects the whole table and can be combined with column widths.

1.3.2 Use more than one header row

A Markdown table normally has one header row. Mark the first additional row with `.header` when a grouped heading needs a second row.

Mark it `{: .header }`:

Markdown

```
| Target {: rowspan=2 } | Measured {: colspan=2 } | | Note {: rowspan=2 } |
| ---|---|---|---|
| | Before {: .header } | After | |
| Widget | 1 | 2 | ok |
```

Result

The rendered table in [Table 24.8](#) shows two rows retained as table headings.

Table 1.8. Use more than one header row

Target	Measured		Note
	Before	After	
Widget	1	2	ok

Both header rows then repeat when a long table continues onto another PDF page.

Widths can be set on either header row. A width on an ordinary cell applies to that physical column even when the cell is in a promoted `.header` row. A width on a merged heading is the total for its `colspan` ; the extension shares that total among the covered columns in proportion to the longest unmerged text in each column. This lets a short identifier and a longer description receive different parts of one grouped width:

```
| Target {: rowspan=2 width="25%" } | Measured values {: colspan=2 width="60%" } |
| Note {: rowspan=2 width="15%" } |
| ---|---|---|---|
| | Before {: .header } | After remediation | |
| Widget | 1 | 2 | ok |
```

Do not put a width on both a merged group and one of its individual columns; those declarations compete for the same space, so the extension reports an actionable error instead of choosing one silently. A grouped width must be a number followed by one CSS unit, such as `60%`, `12rem` or `240px`, so it can be divided without changing its total.

The marker has to go on a cell that **has text** - `attr_list` has nothing to attach to in an empty one. Any cell in the row will do. Only the leading run of marked rows is promoted: a header is the top of a table, and a marked row further down stays where it is rather than the table being quietly re-ordered around it.

1.3.3 Merge cells

Use `colspan` to join cells across columns and `rowspan` to join cells down rows. Keep an empty placeholder for every cell covered by the span, as shown below.

Markdown

```
| Target {: rowspan=2 } | Measured {: colspan=2 } | | Note {: rowspan=2 } |
| ---|---|---|---|
| | Before {: .header } | After | |
| Widget | 1 | 2 | ok |
```

Result

The rendered table in [Table 24.9](#) shows horizontal and vertical cell spans.

Table 1.9. Merge cells

Target	Measured		Note
	Before	After	
Widget	1	2	ok

The empty cell after `Measured` and the empty cells beneath the two `rowspan=2` headings are structural placeholders. The extension removes those placeholders after applying the spans.

For PDF output, all body rows covered by a `rowspan` stay together when the group fits on

one page. This prevents a shaded spanning cell from losing its background when a table continues on the next page; repeated table headers still appear above the moved group.

1.3.4 Fix a table that renders as pipe characters

The header and delimiter rows must declare the same number of cells. A merged heading still needs one empty placeholder for every column it covers:

```
| Target | Risk evaluation {: colspan=3 } | | |
| --- | --- | --- | --- |
```

Without the two empty cells after `Risk evaluation`, Python-Markdown cannot recognise the block as a table at all. `prodockit.tables` detects that failed parse and stops the build with both counts instead of publishing a paragraph of raw pipe characters:

```
row 1 declares 2 cells but the delimiter row declares 4 - a colspan=3 cell
still needs 2 empty placeholder cells after it
```

Fenced and indented code examples are excluded from this check, as is prose that merely contains pipe characters.

1.3.5 Adjust cell shading

Header cells have a subtle 5% shade by default. Remove it from one cell with `shade="off"`, or give any header or body cell an explicit percentage:

Markdown

```
| Unshaded {: shade="off" } | Grouped heading {: colspan=2 shade="8%" } | |
|---|---|---|
| Normal | Highlighted {: shade="5%" } | Normal |
```

Result

The rendered table in [Table 24.10](#) shows shading applied to selected cells.

Table 1.10. Adjust cell shading

Unshaded	Grouped heading	
Normal	Highlighted	Normal

Shading applies to the whole surviving merged cell, so `shade` combines with `colspan` and

`rowspan` on the same attribute list. A percentage must be from `0%` to `100%`; use `off` when the intent is to suppress the default header shade explicitly.

1.3.6 Align content within a tall row

Header and body cells are top-aligned by default on both the website and in the PDF. Mark an individual cell when its content should instead sit in the middle or at the bottom of the row:

Markdown

```
| Detail | Default | At top | Centred | At foot |
| --- | --- | --- | --- | --- |
| First line<br>Second line<br>Third line | Default | Top {: valign="top" } |
Middle {: valign="middle" } | Bottom {: valign="bottom" } |
```

Result

The rendered table in [Table 24.11](#) shows the three distinct vertical positions.

Table 1.11. Align content within a tall row

Detail	Default	At top	Centred	At foot
First line Second line Third line	Default	Top	Middle	Bottom

The accepted values are exactly `top`, `middle` and `bottom`. `valign` applies to the marked header or body cell, including one using `rowspan`, `colspan`, shading, rotation or a promoted header row. Prodockit consumes the authored attribute and emits a stable class, so generated HTML does not retain the obsolete `valign` attribute.

1.3.7 Rotate headings

A wide table is often wide because of its headings, not its data. Turn them on their side:

Markdown

```
| Control | Availability requirement {: rotate=270 width="1.8em"
height="105pt" } |
```

```
| --- | --- |  
| Backups | H |
```

Result

The rendered table in [Table 24.12](#) shows long headings rotated to preserve horizontal space.

Table 1.12. Rotate headings

Control	
Backups	H

`270` reads bottom-to-top, `90` top-to-bottom, and nothing else is allowed: another angle gives a heading nobody can read and a row height nobody can predict.

Set `width` with `rotate` ; a missing width is rejected because rotating text alone does not make the column narrower. Set `height` when a long heading needs more room.

1.4 Reference

Use this section after the worked examples when you need the exact attribute location, accepted value, or generated CSS hook.

1.4.1 Syntax

Builds on Python-Markdown's own `tables` extension (auto-enabled if not already present, the same way [prodockit.refs](#) auto-enables [prodockit.headings](#)).

Separate a cell attribute list from the cell text with at least one space. The extension rejects missing separators, stray tokens and unbalanced quotes in supported table attributes, reporting the source page and line. Fenced and inline code examples, HTML comments and escaped attribute syntax are ignored.

Attach table attributes to header cells. A column's width is a property of the whole column, so declare it once on the heading rather than on a body cell.

[Table 24.13](#) lists every supported attribute and where it belongs.

Table 1.13. Syntax

Attribute	Where to put it	Effect
<code>width = "<css-length>"</code>	Header cell	Set that column's width
<code>.compact</code>	Any header cell	Apply the compact layout to the whole table
<code>.header</code>	A non-empty cell in a leading body row	Move that row into <code><thead></code>
<code>colspan=<n></code>	Cell being widened	Merge it with the following placeholder cells
<code>rowspan=<n></code>	Cell being deepened	Merge it with placeholder cells below
<code>shade="off"</code>	Any cell	Remove shading from that cell
<code>shade="<percentage>"</code>	Any cell	Shade that cell by an explicit percentage
<code>valign = "top" , "middle" or "bottom"</code>	Any cell	Override the default top vertical alignment
<code>rotate =90 or rotate=270</code>	Header cell that also has <code>width</code>	Rotate the heading text
<code>height="<css-length>"</code>	Rotated header cell	Reserve height for the rotated text

The minimal width form is:

```
| Column {: width="<css-length>" } | ... |
|---|---|
```

`<css-length>` is any valid CSS width value - a percentage (`"30%"`) or a fixed length (`"120px"` , `"4cm"` , `"3em"` , ...) - passed through to the generated `<colgroup>` as-is, with no validation of its own; an invalid value behaves exactly as it would in any other hand-written CSS, since `prodockit.tables` doesn't parse or interpret it beyond that.

1.5 Customise with a CSS style sheet

The extension adds stable classes that a website CSS style sheet can target:

[Table 24.14](#) lists the stable table classes available to a custom stylesheet.

Table 1.14. Customise with a CSS style sheet

Element	Condition	Hook
<code><table></code>	at least one header cell has <code>width</code>	<code>class="prodockit-table-sized"</code>
<code><table></code>	any header cell has <code>.compact</code>	<code>class="prodockit-table-compact"</code>
<code><th></code>	heading has <code>rotate=90</code> or <code>rotate=270</code>	<code>class="prodockit-rotate"</code> plus an inline transform
<code><th></code> or <code><td></code>	cell has <code>shade="off"</code>	<code>class="prodockit-table-cell-unshaded"</code>
<code><th></code> or <code><td></code>	cell has <code>shade="<percentage>"</code>	<code>class="prodockit-table-cell-shaded"</code> plus <code>--prodockit-table-cell-shade</code>
<code><th></code> or <code><td></code>	cell has <code>valign="<position>"</code>	<code>class="prodockit-table-cell-valign-<position>"</code>
<code><col></code>	that column has <code>width</code>	<code>style="width: <value>;"</code>

Add at least this rule for sized and compact website tables:

```
.md-typeset table:not([class]),
.md-typeset table.prodockit-table-sized,
.md-typeset table.prodockit-table-compact {
  border-collapse: collapse;
}

.md-typeset table.prodockit-table-sized,
.md-typeset table.prodockit-table-compact {
  table-layout: fixed;
  width: 100%;
  background-color: var(--md-default-bg-color);
  border: 0.05rem solid var(--md-typeset-table-color);
  border-radius: 0.1rem;
  font-size: 0.64rem;
}

:root {
  --prodockit-table-shade-rgb: 0, 0, 0;
}

[data-md-color-scheme="slate"] {
  --prodockit-table-shade-rgb: 255, 255, 255;
}

.md-typeset table:not([class]) th,
.md-typeset table.prodockit-table-sized th,
```

```

.md-typeset table.prodockit-table-compact th {
  background-color: rgba(var(--prodockit-table-shade-rgb), 0.05);
}

.md-typeset table th.prodockit-table-cell-unshaded,
.md-typeset table td.prodockit-table-cell-unshaded {
  background-color: transparent;
}

.md-typeset table th.prodockit-table-cell-shaded,
.md-typeset table td.prodockit-table-cell-shaded {
  background-color: rgba(
    var(--prodockit-table-shade-rgb),
    var(--prodockit-table-cell-shade)
  );
}

.md-typeset table:not([class]) th,
.md-typeset table:not([class]) td,
.md-typeset table.prodockit-table-sized th,
.md-typeset table.prodockit-table-sized td,
.md-typeset table.prodockit-table-compact th,
.md-typeset table.prodockit-table-compact td {
  vertical-align: top;
}

.md-typeset table th.prodockit-table-cell-valign-middle,
.md-typeset table td.prodockit-table-cell-valign-middle {
  vertical-align: middle !important;
}

.md-typeset table th.prodockit-table-cell-valign-bottom,
.md-typeset table td.prodockit-table-cell-valign-bottom {
  vertical-align: bottom !important;
}

.md-typeset table:not([class]) th,
.md-typeset table:not([class]) td,
.md-typeset table.prodockit-table-sized th,
.md-typeset table.prodockit-table-sized td,
.md-typeset table.prodockit-table-compact th,
.md-typeset table.prodockit-table-compact td {
  border: 0.05rem solid var(--md-typeset-table-color);
}

```

The complete maintained light- and dark-mode rules used by this site are in `docs/stylesheets/pdk.css` . Add project-specific changes to `docs/stylesheets/extra.css` . PDF builds include equivalent layout rules with the light website scheme's line colour and the

same width; use `docs/stylesheets/print.css` for PDF-only overrides. Contributors changing the generated table structure should read [Extension integration](#).