Table of Contents

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, 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:

Need	Attribute
Set a column width	`width="30%"` or a fixed width such as `8rem`
Fit many short columns	`.compact`
Repeat more than one header row	`.header`
Merge cells	`colspan=2` or `rowspan=2`
Turn a long heading vertically	`rotate=90` or `rotate=270`, with `width`

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

1.2 Set column widths

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

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

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

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

#	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

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

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

Target	Measured		Note
	Before	After	
Widget	1	2	ok

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

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

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.

1.3.4 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

Control	Availability requirement
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

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).

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.

Attribute	Where to put it	Effect
`width="<css-length>"`	Header cell	Set that column's width
`.compact`	Any header cell	Apply the compact layout to the whole table
`.header`	A non-empty cell in a leading body row	Move that row into `<thead>`
`colspan=<n>`	Cell being widened	Merge it with the following placeholder cells
`rowspan=<n>`	Cell being deepened	Merge it with placeholder cells below
`rotate`=90 or `rotate=270`	Header cell that also has `width`	Rotate the heading text
`height="<css-length>"`	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:

Element	Condition	Hook
`<table>`	at least one header cell has `width`	`class="prodockit-table-sized"`
`<table>`	any header cell has `.compact`	`class="prodockit-table-compact"`
`<th>`	heading has `rotate=90` or `rotate=270`	`class="prodockit-rotate"` plus an inline transform
`<col>`	that column has `width`	`style="width: <value>;"`

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

```
.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;
}
```

The complete light- and dark-mode rules used by this site are in `docs/stylesheets/extra.css`. PDF builds already include equivalent layout rules. Contributors changing the generated table structure should read Extension integration.

Index

P