Table of Contents

1. Tables

`prodockit.tables` gives a table column a percentage or fixed width, via a `width` attribute already attachable to a header cell with `attr_list` - no new syntax to learn.

1.1 Quick start

Enable it in `zensical.toml`:

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

then attach a `width` to any header cell you want to size.

1.1.1 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

1.1.2 Percentages that don't add up to 100%

Leave a column unannotated and it takes whatever's left over, split evenly if more than one column is left unannotated - the same "standard algorithm" any HTML

table with a `<colgroup>` and `table-layout: fixed` already uses to size an unspecified column, not something `prodockit.tables` computes itself:

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>`.

1.1.3 Fixed widths for every column

A CSS length works the same way as a percentage - useful for a column that should stay a constant size regardless of how wide the table's container ends up being. Every column can be given one, with none left to share remaining space:

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

1.1.4 Mixing percentages and fixed widths

Percentage and fixed-length columns can appear in the same table - CSS's own table layout algorithm sizes both correctly at once, the same way it would for any hand-written `<colgroup>`. A column can still be left unannotated here too, sharing whatever's left over the same way as the percentage-only case above:

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

1.1.5 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.

1.2 Reference

1.2.1 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 `width` to a header cell (`th`), not a body cell - a column's width is a property of the whole column, so it's declared once, on the heading:

```
| 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.

1.2.2 CSS hooks

A table with at least one `width`-attributed header cell gets a `<colgroup>` (one `<col>` per column, `style="width: ..."` set only on the columns that had one) inserted as its first child, and `class="prodockit-table-sized"` on the `<table>` itself:

Element	Condition	Hook
`<table>`	at least one header cell has `width`	`class="prodockit-table-sized"`
`<col>`	that column's header cell had `width`	`style="width: <value>;"`
`<col>`	that column's header cell had no `width`	none - left for `table-layout: fixed` to size

The `width` attribute itself is always stripped from the `<th>` once read - it isn't meant to also linger on the header cell.

`prodockit-table-sized` only *marks* a table as sized - it isn't styled by `prodockit.tables` itself. A stylesheet needs to apply `table-layout: fixed` itself for the `<colgroup>` widths (and the "share what's left" behaviour) to take effect at all. Under Zensical's Material-based theme specifically, its own default table styling (border, padding, alternating rows) is scoped `.md-typeset table:not([class])` - confirmed directly in Zensical's bundled CSS - so a table carrying `prodockit-table-sized` (or any other class) gets **none** of it, not just no width control:

```
.md-typeset table.prodockit-table-sized {
  table-layout: fixed;
  width: 100%;
  border-collapse: collapse;
  border: 1px solid #555555;
}
.md-typeset table.prodockit-table-sized th,
.md-typeset table.prodockit-table-sized td {
  border: 1px solid #555555;
  padding: 0.9375em 1.25em;
  vertical-align: top;
```

```
}
.md-typeset table.prodockit-table-sized th {
  font-weight: 700;
}
```

prodockit.pdf already includes the equivalent rule in its own generated CSS (as well as its own table border/padding, unaffected by this theme-specific quirk), so a sized table works in the PDF with no extra configuration; a project's own website theme needs the CSS above added itself (see this project's own `docs/stylesheets/extra.css` for a working example) - `prodockit.tables` doesn't ship a bundled website stylesheet the way `prodockit.pdf` ships one for the PDF path.

A table with no `width`-attributed header cells at all is left completely untouched - no `<colgroup>`, no `prodockit-table-sized` class - so enabling `prodockit.tables` has no effect on any table that doesn't use it.