

# Table of Contents

|                                                              |   |
|--------------------------------------------------------------|---|
| <b>1. Authoring reference</b> .....                          | 2 |
| 1.1 Build on PyMdown Blocks .....                            | 2 |
| 1.2 Follow the same three stages .....                       | 2 |
| 1.3 Choose a feature .....                                   | 3 |
| 1.3.1 Choose a supported alternative when it is enough ..... | 3 |
| 1.4 Use features beyond Markdown extensions .....            | 5 |
| 1.5 Keep deployment concerns separate .....                  | 5 |

# 1. Authoring reference

This section is for a document author who has built a first Zensical site and wants to add structure or specialist content to its Markdown pages, use values calculated across the document, or produce a PDF. You do not need to read every page: choose the feature or tool the document needs and start with its smallest complete example.

[Figure 19.1](#) starts with the document's need and groups the available features into three paths. Use the top path for structure and layout, the middle path for evidence and navigation, and the bottom path for calculated values or generated outputs.

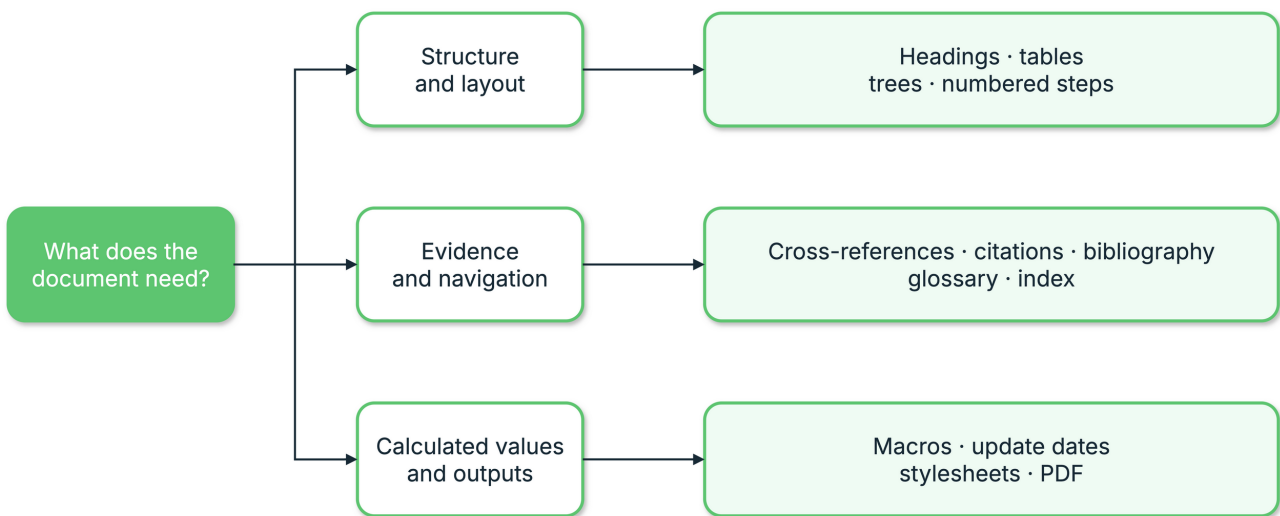


Figure 1.1. Prodockit authoring feature map

## 1.1 Build on PyMdown Blocks

Two prodockit extensions are built directly on PyMdown Blocks (see the [upstream Blocks guide](#)): `prodockit.steps` for procedures and `prodockit.tree` for directory listings. They use PyMdown's slash-fenced container syntax, nesting rules, and block options rather than introducing a separate container language.

If you already use PyMdown Blocks, the structure will be familiar. If you do not, start with [Numbered steps](#): its first example shows the complete opening and closing fences before explaining nested blocks.

## 1.2 Follow the same three stages

Every extension guide starts with the same learning path:

### 1. Enable the extension

Add the extension's table to `zensical.toml`. Each prodockit extension is independent, so enabling headings does not silently enable citations, tables, or another feature.

### 2. Write the Markdown

Start with the complete copyable example. The guide shows both its Markdown source and rendered result before introducing variations.

3. Configure only when needed

Keep the defaults for a first use. The configuration section explains the available `zensical.toml` settings and shows a rendered example when an option changes visible output.

1.3 Choose a feature

Match the document outcome you need to its authoring feature in [Table 19.1](#).

Table 1.1. Choose a feature

| Document need                                                           | Reference                                             |
|-------------------------------------------------------------------------|-------------------------------------------------------|
| Number sections and give headings stable links                          | <a href="#">Headings</a>                              |
| Refer to a heading, figure, or table without typing its changing number | <a href="#">Cross-references</a>                      |
| Define short references directly in Markdown                            | <a href="#">Hand-written citations and references</a> |
| Expand abbreviations and define specialist terms                        | <a href="#">Acronyms and glossary</a>                 |
| Control column widths, merged cells, and dense layouts                  | <a href="#">Tables</a>                                |
| Show a folder and file hierarchy                                        | <a href="#">Directory trees</a>                       |
| Present a procedure with connected numbered stages                      | <a href="#">Numbered steps</a>                        |
| Cite a BibTeX library in a selected CSL style                           | <a href="#">Bibliography</a>                          |
| Mark terms for a PDF back-of-book index                                 | <a href="#">Index</a>                                 |

The two citation choices in [Table 19.1](#) are approaches to the same broad task. A small document can define sources directly in Markdown; a report with an existing `.bib` library normally uses the bibliography extension. You do not need to enable both.

1.3.1 Choose a supported alternative when it is enough

Prodockit extends rather than replaces the features Zensical already supports. Before enabling another extension, check Zensical's current [MkDocs plugin compatibility](#) and [Python-Markdown extension compatibility](#) lists. [Table 19.2](#) identifies the closest supported alternative to each Prodockit extension and the narrower case where that alternative is sufficient.

Table 1.2. Supported alternatives to Prodockit extensions

| Prodockit feature                   | Closest supported alternative                                                                         | Use the supported alternative when...                                                                                                                                                  |
|-------------------------------------|-------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>prodockit.headings</code>     | Python-Markdown <a href="#">toc</a> and <a href="#">attr_list</a>                                     | You need stable heading ids and permalinks, but not hierarchical numbering across pages, appendices, captions, or navigation.                                                          |
| <code>prodockit.refs</code>         | Zensical <a href="#">autorefs</a>                                                                     | You want path-independent links and will write their visible text yourself; you do not need automatic section numbers, figure or table labels, or PDF page numbers.                    |
| <code>prodockit.citations</code>    | <a href="#">Footnotes</a> and ordinary links                                                          | A source note belongs only to its current page and does not need a reusable key, generated citation text, or a cross-page reference entry.                                             |
| <code>prodockit.glossary</code>     | <a href="#">abbr</a> , <a href="#">attr_list</a> , and <a href="#">pymdownx.snippets</a>              | A central set of browser tooltips is enough; you do not need explicit <code>\gls{id}</code> links, missing-term markers, or separate linked acronym and glossary pages.                |
| <code>prodockit.tables</code>       | Standard <a href="#">data tables</a> , <a href="#">attr_list</a> , and <a href="#">Blocks Caption</a> | You need an ordinary table or caption, but not controlled widths, dense layout, multiple header rows, merged cells, shading, vertical alignment, or rotated headings.                  |
| <code>prodockit.steps</code>        | A Markdown numbered list                                                                              | The sequence is short and does not need titled connected stages, rich nested block content, continued numbering, or matching website and PDF presentation.                             |
| <code>prodockit.tree</code>         | A Markdown list or fenced code block                                                                  | A plain hierarchy is sufficient and you do not need indentation validation, file/directory recognition, descriptions, or configurable icons.                                           |
| <code>prodockit.bibliography</code> | <a href="#">Footnotes</a>                                                                             | You do not need a BibTeX/BibLaTeX library, CSL formatting, keyed citations, or an automatically generated reference list. There is no direct equivalent in Zensical's supported lists. |
| <code>prodockit.index</code>        | Zensical <a href="#">search</a> or <a href="#">tags</a>                                               | Readers will use the website only; you do not need an alphabetised PDF back-of-book index with page numbers and hierarchical entries.                                                  |

The alternatives in [Table 19.2](#) overlap with a specific part of each feature; they are not drop-in replacements for the whole row. Prefer the smaller supported feature when its stated boundary matches the document, and use the Prodockit extension when the document needs the additional behaviour.

## 1.4 Use features beyond Markdown extensions

Some authoring features are commands or Zensical template helpers rather than Python-Markdown extensions:

[Table 19.3](#) identifies the authoring features provided by commands or template helpers rather than Markdown syntax.

Table 1.3. Use features beyond Markdown extensions

| Document need                                                                                      | Reference                          |
|----------------------------------------------------------------------------------------------------|------------------------------------|
| Insert calculated values such as word counts, repository details, or document-wide layout settings | <a href="#">Website macros</a>     |
| Show when each source page was last updated                                                        | <a href="#">Page update dates</a>  |
| Produce a complete PDF, a single-page PDF, or a source bundle                                      | <a href="#">PDF generation</a>     |
| Find the safe first form, write behaviour, and options for every public command                    | <a href="#">Command-line tools</a> |

The non-extension features in [Table 19.3](#) are part of the same authoring reference because they affect what the document contains or produces. Installation, repository maintenance, continuous integration, and deployment remain in their task-based sections.

## 1.5 Keep deployment concerns separate

The authoring pages explain what to write, how to configure its rendered features, and how to build local outputs. When those outputs are ready for a hosted website, continue to [Publish a document](#). Template updates, CI, Pages deployment, and output tests live there so they do not interrupt the authoring reference.