Table of Contents

1. Hand-written citations and references

`prodockit.citations` lets you write a reference once and cite it from any page. A citation such as `[Skoulikari, 2023]` links readers to the full reference.

> **Looking for an auto-generated bibliography instead?**
>
> This page is for a short reference list that you write yourself. To create a reference list automatically from a `.bib` file, use Bibliography.

Use this extension when you want to write and format a short reference list by hand, but define each citation's link text only once.

1.1 Enable the extension

Enable it in `zensical.toml`:

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

1.2 Add and cite a reference

Write the full reference, then add its id and the shorter text you want to show in citations on the line below. Cite it with `\citeref{id}`:

Markdown

```
Git is a tool used to manage version control.\citeref{skou-
example}

Skoulikari, A. (2023) *Learning Git: A Hands-On and Visual Guide
to the
Basics of Git*. Sebastopol, CA: O'Reilly Media.
{: #skou-example .reference data-cite-text="Skoulikari, 2023" }
```

Result

Git is a tool used to manage version control.[Skoulikari, 2023]

Skoulikari, A. (2023) *Learning Git: A Hands-On and Visual Guide to the Basics of Git*. Sebastopol, CA: O'Reilly Media.

The extension replaces `\citeref{skou-example}` with the linked citation `[Skoulikari, 2023]`. Select it to jump to the full reference.

1.3 Configure citations

1.3.1 Choose the missing-citation text

An unresolved citation displays `?` by default. Set `unresolved` if your project uses a different marker:

```
[project.markdown_extensions."prodockit.citations"]
unresolved = "MISSING"
```

`source` is the only other TOML setting. It identifies the current page, but Zensical detects it automatically; leave it unset in `zensical.toml`.

Multiple comma-separated keys join into one bracket: `\citeref{skou2023,chacon2014}` → `[Skoulikari, 2023; Chacon and Straub, 2014]`.

1.3.2 Cite a source before its full reference

A citation to a source defined *later* in the same document resolves correctly:

Markdown

```
See \citeref{forward-citation-example} for an introduction to
Git.

Skoulikari, A. (2023) *Learning Git*.
{: #forward-citation-example data-cite-text="Skoulikari, 2023" }
```

Result

See [Skoulikari, 2023] for an introduction to Git.

Skoulikari, A. (2023) *Learning Git*.

1.3.3 Fix a missing citation

If a citation id is missing or mistyped, the extension displays `?` for that entry. Other valid entries in the same citation still work:

Markdown

```
\citeref{forward-citation-example,does-not-exist}
```

Result

[Skoulikari, 2023; ?]

Check the id that produced `?` against the id on the full reference.

1.4 Reference

1.4.1 Syntax

Defining and citing are bundled into one extension, unlike prodockit.headings/ prodockit.refs: a definition is useless without somewhere to cite it, so there's no independently useful "just defining" half to split out.

Defining a source

Any block element - typically a paragraph - with both an `id` and a `data-cite-text` attribute becomes a citable source:

```
Chacon, S. and Straub, B. (2014) *Pro Git*. 2nd edn. New York:
Apress.
{: #chacon2014 .reference data-cite-text="Chacon and Straub, 2014" }
```

`data-cite-text` is the short text rendered at each citation site - it's stripped from

the rendered output (it's internal bookkeeping, not meant to be visible), while `id` stays, since citations link straight to it.

Citing a source

Syntax	Purpose
`\citeref{<id>}`	Cite one defined source
`\citeref{<id1>,<id2>,...}`	Cite several sources in one bracket

Like prodockit.refs, `\citeref{...}` is recognised the same way Python-Markdown's own inline syntax is, so it's protected inside inline code spans and fenced code blocks:

````
Type `\citeref{skou2023}` to cite a source.

```
\citeref{skou2023}
```
````

Neither of the two shown above is resolved; both render the literal text.

1.4.2 Zensical settings

Setting	Default	What it controls
`unresolved`	`"?"`	Text shown for a citation id that cannot be found.
`source`	`""` (detected automatically)	Advanced: identifies the current page when using the extension outside Zensical. Leave it unset in `zensical.toml`.

1.4.3 Cross-page citations

Under Zensical, a citation can refer to a source defined on another page, including a references page later in navigation. Prodockit reads definitions across the navigation before page conversion, so no author configuration is required.

Two definitions using the same key produce a warning and the first definition is retained. Use a unique key for every source.

For integration with another Markdown renderer, see Extension integration.

1.4.4 What this doesn't do (yet)

`prodockit.citations` covers citation-key management - the "define once, cite

anywhere" part. It doesn't auto-generate the references page's listing itself from structured bibliographic data (author/year/title/publisher/URL fields) the way a full BibTeX-style tool would - the reference entry's own text (as shown in the examples above) is still hand-authored prose, just like today. See prodockit.bibliography for the alternative that does exactly this, from a `.bib` file, in any citation style - and for the tradeoffs between the two approaches.

1.5 Customise with a CSS style sheet

`prodockit.citations` emits three hooks - one on the outer wrapper, one on each individual key's own link:

Element	State	Class
Outer `<span>` wrapping the whole `\citeref{...}` citation	always	`prodockit-cite`
Each key's own `<a>`	resolved	`prodockit-cite-resolved`
Each key's own `<a>`	unresolved	`prodockit-cite-unresolved`

An unresolved key's `<a>` has no `href` (see Unresolved citations above) - style `prodockit-cite-unresolved` distinctly (e.g. a muted colour, no underline) to make a missing citation visually obvious.

Index

P