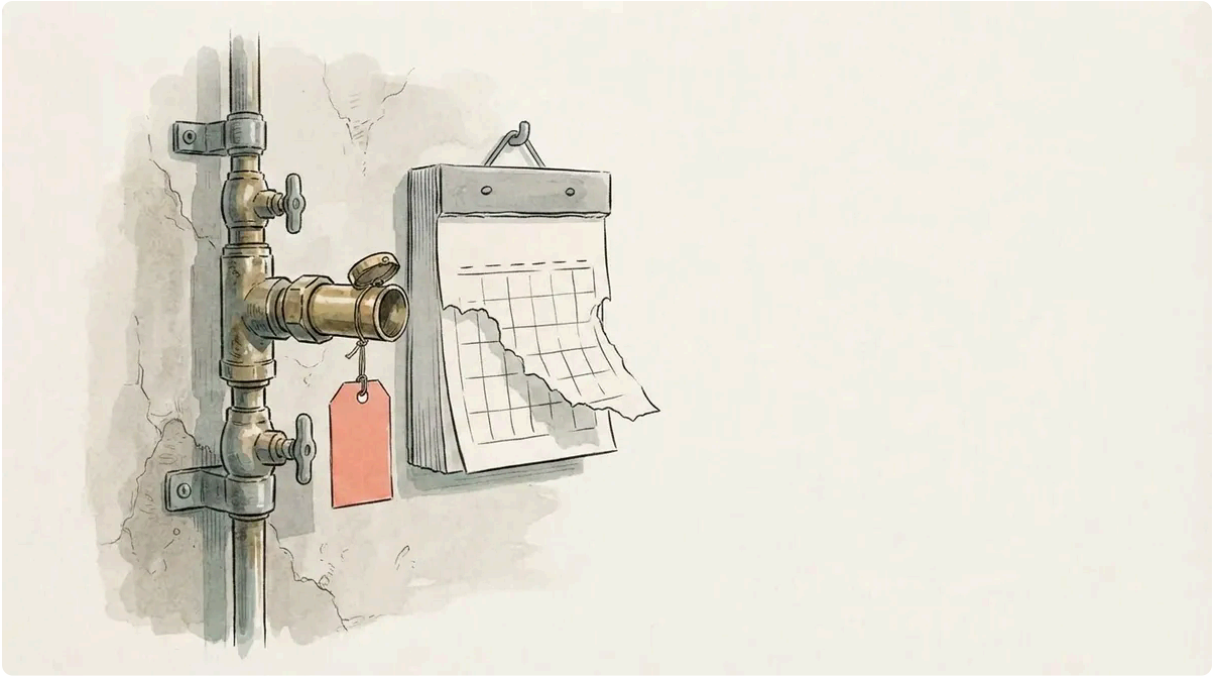
# Citations by DOI, and the warning I read past for a month

Emil Lindfors | September 29, 2026



A DOI is a string. `10.1016/j.marpol.2016.10.020` is one, and the thing that turns it into a reference is an HTTP request. The citations on this site used to need a desktop application for that, and for a month this summer they needed nothing at all, because the pipeline was dead and the build was hiding it. Line 26 of `build.sh`:

```
"$SITE_TOOLS" cite all || echo "  Warning: citation processing failed"
```

That line printed its warning on every build for a month, one line in the middle of about forty, and I read past it every time. I only found out when I went to add a citation to a post and the citekey came out the other side unchanged. No error, no stack trace, just `@Christiansen2017` sitting in the rendered HTML exactly as I had typed it.

The warning was telling the truth. Here is what it was hiding, and what the pipeline looks like now that it runs on DOIs.

## What actually broke

Better BibTeX migrated its store on 27 July. `~/Zotero/better-bibtex.sqlite` became `~/Zotero/better-bibtex.migrated`, which still holds the `citationkey` table it always did. `zotero-cite` opens `default_bbt_db()` at the old path, gets `unable to open database file`, and returns an error. `build.sh` catches that, prints it, and steps over it.

The failure was a plugin doing something entirely reasonable during an upgrade. The damage was a `||` I had written months earlier to keep the build going on a machine without Zotero installed. Nothing was corrupted, because nothing was processed. If your build script has a `||` in it, go and look at what it hides.

## Where the keys went

I spent a while assuming they were gone. `zotero.sqlite` has 158 objects in it and not one of them is BBT-shaped. `~/Zotero/better-bibtex/` contains a single 2-byte `read-only.json`. There isn't even a Zotero profile directory under `AppData` on this machine. This paragraph said "somewhere I have not found yet" for most of a day.

They were in `zotero.sqlite` the whole time. Zotero 8 added a native `citationKey` field, and the migration is Better BibTeX handing its keys over to it:

```
select v.value from itemData d
  join itemDataValues v on v.valueID = d.valueID
  join fields f on f.fieldID = d.fieldID
 where f.fieldName = 'citationKey';
```

1,421 rows. `Christiansen2017` is one of them. I had gone looking for a table with a plugin's name on it, and this was a schema change with the plugin stepping out of the way.

So the fix is smaller than the problem looked. `site-tools` reads `zotero.sqlite` directly now. The citekey sits in `itemData` next to the title and the DOI, one join away from the creators, and both the Better BibTeX dependency and the crate that used to wrap it are gone. The database opens read-only with `immutable=1`, which skips SQLite's locking protocol so a build can read the library while Zotero has it open. That also means a build cannot write to it. That is the guarantee I want from something a script runs unattended.

## The Zotero dependency

Part one describes a pipeline that reads citation keys out of Zotero's SQLite database at build time. I still think that design is right for what it is. But look at what it needs: a desktop application, installed on a particular machine, with a particular plugin, keeping a database at a particular path, in a schema neither of us controls.

For a static site whose whole premise is text files in a git repo, that is a strange thing to require. I can rebuild this site from a clone on any machine. Except the citations, which only work on the laptop with Zotero on it.

A DOI needs none of that. It is a string in the post, and one HTTP request turns it into a reference.

## The crossref client

I had one to hand. That is most of why this happened now and not in six months.

crossref-client is a hard fork of crossref-rs. crossref-rs is a good crate and is no longer maintained. I wanted a CLI, the search side of it was thin, and there were a few other gaps. Nobody was reading pull requests there, so I started my own thing. It's async now, targets Rust 2024, and the query and response types have moved far enough that none of it is upstreamable even if someone picked the original back up.

Nobody has used it. I haven't announced it and it was really an internal tool, so take what follows as a report on something that works for me and not as a recommendation.

The API behind it is really good. Crossref documents what it serves, tells you your remaining request budget in a response header (the client reads that and slows itself down instead of guessing), and keeps a public list of what is changing and what is coming. That last one is rarer than it should be. I have spent a lot of hours reverse-engineering APIs from their error messages, and being told in advance is a small luxury.

## What a citation looks like now

Two forms, following pandoc closely enough that nobody has to learn a third convention:

```
Research by @Christiansen2017 found it, and [@10.1080/13657305.2017.1262476]
agrees.
Both [@Christiansen2017; @10.1080/13657305.2017.1262476] say the same.
```

`@key` is narrative and carries the sentence. `[@key]` is parenthetical, and several join with a semicolon. A key that is a DOI resolves against crossref directly. Anything else is looked up in a map in the post's own frontmatter:

```
[extra.bib]
Christiansen2017 = "10.1016/j.marpol.2016.10.020"
```

A DOI has to be bracketed. It contains dots and a slash, so a bare `@10.1016/j.marpol.2016.10.020.` at the end of a sentence has no way to tell whether that last full stop belongs to it. I spent a while trying to be clever about this and then stopped.

Brackets settle it, and a source you cite often enough to want narratively is a source worth giving a name.

The tool says so if you forget:

```
post: @10.1080/13657305.2017.1262476 looks like a DOI. Write it as
[@10.1080/13657305.2017.1262476] -- a bare one cannot tell where it ends.
```

## Keeping Zotero, without configuring anything

The obvious way to support two sources is a config setting. I didn't want one, because a setting is a thing you have to know about before it can help you.

The marker's shape picks the source instead. A DOI, or a key the post's `[extra.bib]` maps to one, goes to crossref. A key with no entry there falls back to the Zotero library, exactly as before. A site that only ever writes DOIs never opens a Zotero database, and someone who prefers their collection carries on writing citekeys and never touches the network. There's a `--source crossref|zotero|auto` flag for when the routing guesses wrong, and I have not needed it yet.

This matters more than the tidiness of it. The journal literature I cite here is exactly what crossref is good at: Christiansen & Jakobsen (2017) on how the Norwegian salmon industry narrates its own greening resolves from a DOI in about a second. Books, technical reports, standards and most of the grey literature I work from in aquaculture have no DOI at all. Crossref covers journal articles superbly and everything else patchily. Zotero is the half that handles what crossref can't.

## Where the references live

Resolved records go into `[[extra.references]]` in the post's own frontmatter:

```
[[extra.references]]
key = "Christiansen2017"
type = "article"
author = "Christiansen, E. A., & Jakobsen, S. E."
title = "Diversity in narratives to green the Norwegian salmon farming industry"
year = "2017"
journal = "Marine Policy"
volume = "75"
pages = "156-164"
doi = "10.1016/j.marpol.2016.10.020"
```

- ` elements to the bottom of the file. I don't know when it drifted. The post described the design I wanted, the code had been doing something else, and I had stopped being able to tell which was which.

The templates were ready either way. `templates/components.html` has had a `bib.reference` component since the Zola 0.23 migration, branching on entry type for articles, books, conference papers and theses. Nothing had ever populated it.

Storing the record in the post is what makes the thing offline. After the first run there are no markers left to resolve, so a build touches neither crossref nor Zotero, a re-run comes back byte-identical, and the reference can't drift from the post citing it.

## Three snags

**Zola failed the build on my own anchors.** An inline citation links to its entry: `2017`. Zola resolves that fragment against the page's own content and errors if it can't find the target. The reference list is now rendered by a template from frontmatter, so its `id` attributes are not in `page.content`, and every citation in the post became a broken internal link as far as the build was concerned.

The fix is to write the anchor as raw HTML, `2017`, which passes through unchecked. That is a grubby thing to have in a markdown source and I went looking for a better answer before accepting it. Nothing is lost by the check going away, because a marker is only ever rewritten once its reference has been stored, so the target exists by construction. It did mean the PDF, the plain-markdown representation and the text-to-speech script all had to learn to strip an HTML anchor instead of a markdown link.

**Crossref returns titles as XML fragments.** `Aquaculture Economics &amp; Management`. Also `<i>Salmo salar</i>` where a publisher used markup in a title. The templates escape what they render, so an `&` left in place renders as the five characters `&amp;` on the page. Entities have to be decoded and tags stripped before storing, tags first, or a `&lt;` decoded early gets read as the start of a tag on the next pass.

Two smaller ones from the same family. Crossref names a publisher on every journal article, so every reference carried a `publisher = "Elsevier BV"` line that no template prints. And it sets `url` to the DOI resolver you already have, so each entry rendered a `doi:` link and a `[link]` pointing at the same place.

**The reference list reordered itself on the second run.** I ran the tool twice on the same post to check it was idempotent and the two references swapped places. The function that reads existing references back out of frontmatter returned a `BTreeMap`, so the order was alphabetical by anchor, and `10-1080-13657305-2017-1262476` sorts above `Christiansen2017`. It returns a `Vec` in file order now, with a test that says so. Re-sorting a published post's reference list on every run is how you get a baffling diff six months later.

## The post that nearly rewrote itself

Part one is a post about citation syntax, so it is full of `@citekey` examples inside code fences. The processor is not code-aware. It has an opt-out, `skip_citations = true` in the frontmatter, and I set that flag back in February with a comment explaining why.

Checking it properly for the first time this week: `@Christiansen2017`, `@osmundsenFishFarmersRegulators2017` and `@iversenProductionCostCompetitiveness2020` all appear inside code blocks in that post, and all three are real entries in my library. There are 1,421 keys in it. Those three were one lookup away from being rewritten into rendered citations inside the code samples that exist to show you what the unrendered syntax looks like, with a references section appended underneath for good measure, and `deploy.sh` would have committed and pushed the result.

So the mask came first. Fenced blocks, inline code spans and the TOML frontmatter come out before the processor sees the text and go back in afterwards, as placeholders wrapped in NUL bytes. Indented code blocks are deliberately not masked: four leading spaces mean "code" outside a list and "continuation" inside one, and guessing wrong would hide real prose from the processor, which fails silently. Every code block in this repo is fenced.

`skip_citations` is gone from part one, and running the tool over it now returns the file byte for byte.

## Still open

Whether crossref's coverage is enough. Everything I've cited on this blog so far has a DOI, which is not a coincidence, because I cite journal articles here and leave the reports in a drawer. The first time I want to cite a Directorate of Fisheries statistics release I'll find out how much of the old path I still need.

The reference formatting is still hardcoded in Tera components. That was on the "what I'd change" list in part one, and it still is. Crossref will render a citation in any of about 2,900 CSL styles server-side, and `transform(doi, CnFormat::bibliography("apa"))` is the whole call, so switching between APA and IEEE went from a formatting engine to a one-line lookup. That lookup is `site-tools cite format <doi> --style ieee` now, and it prints a formatted reference in about a second. The list on the page still comes from the structured fields, because crossref cannot format a reference that only Zotero knows about, and a list assembled by two formatters would show the seam.

And the two sources disagree in small ways I haven't decided about. Crossref gives Christiansen and Jakobsen no issue number at all; Zotero has `October 2016` sitting in the issue field for the same paper. Neither is wrong, and both come from whatever the publisher deposited. But a post citing one source through crossref and another through Zotero

ends up with a reference list assembled to two slightly different sets of rules, and I don't know yet whether that will bother me enough to normalise it.

If you want to try this on your own site, the client is on GitHub. Write DOIs in brackets, give a name in `[extra.bib]` to anything you cite more than once, and read your build output for the word Warning.

## References


- Christiansen, E. A., & Jakobsen, S. E. "Diversity in narratives to green the Norwegian salmon farming industry". *Marine Policy*, vol. 75, pp. 156-164, 2017. doi:10.1016/j.marpol.2016.10.020