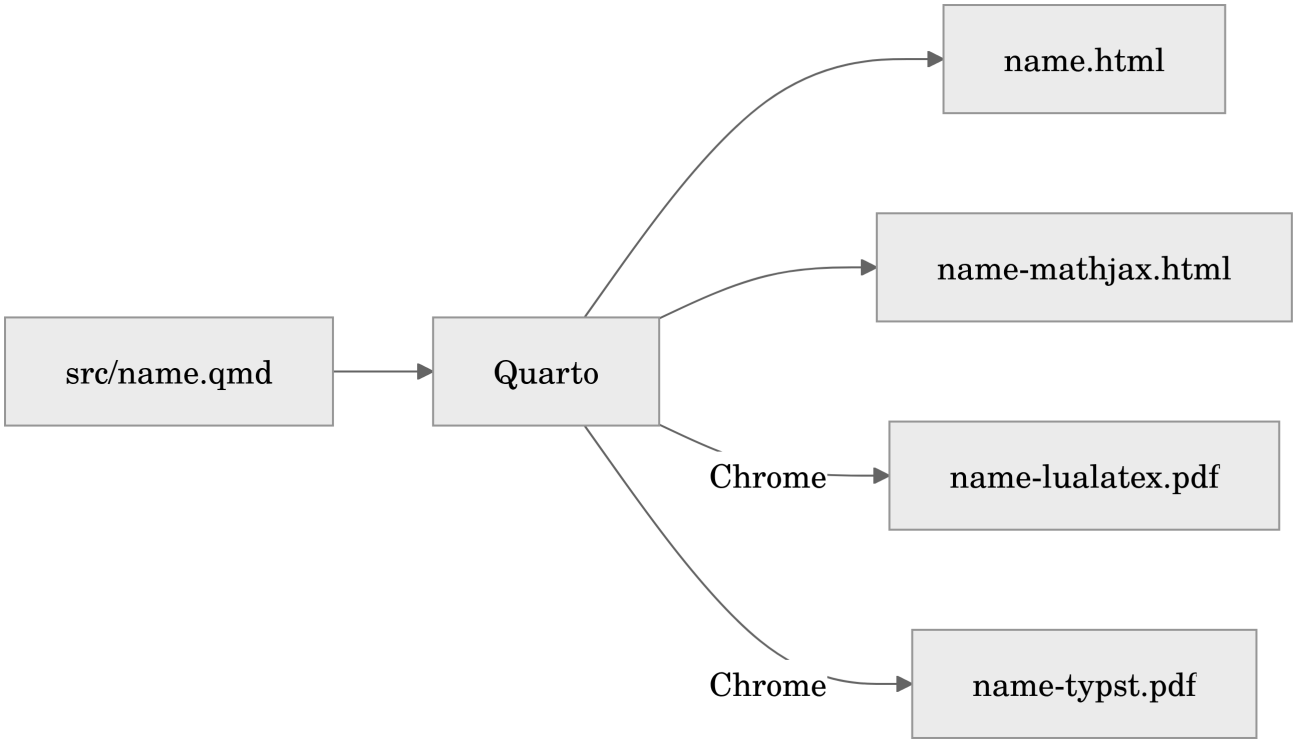
Diagrams through Quarto's own machinery

Table of contents

This is the second of the two diagram patterns in this repository, and the shorter one to set up. It uses Quarto's own mermaid support and writes no filter at all. The other one renders each diagram ahead of time and substitutes it with a Lua filter, which is more machinery for a picture that works in twice as many outputs.

Both put the labels in TeX Gyre Schola. They differ in what that costs and in what you get back.



What makes this page different from every other source here

It is a `.qmd`, and it is the only one. That is not a stylistic choice: the diagram above is an executable cell, and Quarto refuses those in a `.md`.

```
ERROR: You must use the .qmd extension for documents with executable code.
```

The extension is the smaller half of the consequence. The larger half is that **this page has no vanilla Pandoc rendering at all**, so it appears in four outputs where the other sources appear in eight. The Makefile is not missing it by oversight — it globs `src/*.md`, and this file is deliberately outside that glob, because Pandoc does not merely ignore an executable cell, it misreads one. Given

````
```{mermaid}
flowchart LR
  A ⟶ B
```
````

Pandoc's attribute parser rejects the dotless `{mermaid}`, so the fence is never recognised as a code block at all and the whole diagram collapses into a single run-on inline code span inside a paragraph:

```
Para [ Code ("",[],[]) "{mermaid} flowchart LR   A ⟶ B" ]
```

A missing figure would be survivable. A mangled paragraph in the middle of the prose is not, which is why this file is kept out of the Makefile rather than rendered and inspected.

The same misparse keeps it out of `pixi run format`, whose pattern is also `*.md`. That round trip reads and rewrites each source through Pandoc, so running it over this file would corrupt the cell permanently. This page is therefore not covered by CI's `format-check` either.

Writing the block as ```` ```{.mermaid} ```` — the form Pandoc does understand — does not bridge the two. Quarto executes only the dotless form, and emits the dotted one as a literal code listing, `%%{init}%%` directive and all. The two syntaxes are mutually exclusive, and that, rather than the file extension, is what makes this a separate pattern instead of a setting.

Setting the font, twice

The font is named in two places, and neither one covers the other.

Output	Rendered by	Where the family is named
both HTML	mermaid.js, in the reader's browser	`--mermaid-font-family`, in `config/mermaid-schola.html`
both PDF	mermaid in headless Chrome, at build time	`%%{init}%%` in the diagram source

The HTML half cannot use the `%%{init}%%` directive. Mermaid writes those theme variables into a stylesheet inside the SVG it builds, and Quarto's own mermaid stylesheet — which sets `font-family: var(--mermaid-font-family)` on every label — overrides it. The directive is not rejected; it simply has no visible effect, which is the least helpful way for a setting to fail.

The PDF half is the reverse. It renders in a bare headless-Chrome page that never sees the site's CSS, so the custom property means nothing there and the directive in the diagram is the only thing that names the face.

What it costs

`mermaid-format: png` is not optional, and forgetting it is silent. With the default, both PDF recipes render successfully, exit zero, and simply do not contain the diagram. `mermaid-format: svg` behaves the same way. Only `png` produces a picture, which means the PDFs get a **raster** — 383 ppi here, which prints acceptably but is not selectable, not searchable, and not resolution-independent. The other pattern puts real text in an embedded Schola subset into both PDFs.

The build needs a browser. Quarto renders those PNGs through a headless Chrome it installs itself, and on a machine that has never done so the render stops:

```
ERROR: Chrome not found
No Chrome or Chromium installation was detected.
Please run 'quarto install chrome-headless-shell' to install a headless browser.
```

That is a 262 MB download, and CI does it on every run. It fails loudly, at least, which the two silent failures above do not.

What it buys

No generated files in the repository, and nothing to keep in sync. There is no Lua filter, no render script, no checked-in artifact, and no staleness to guard against, because the picture is drawn from the source on every render.

The HTML diagrams are live and they follow the theme. They are laid out in the reader's browser rather than baked at author time, so they re-flow with the page and pick up the light and dark themes instead of being a fixed-colour plate — which is the one thing the other pattern cannot do.

Choosing between them

	This page	The Lua filter
Outputs	4, Quarto only	8, both pipelines
Source extension	`.qmd`	`.md`
PDF diagram	raster PNG	vector, embedded Schola subset
HTML diagram	live, follows the theme	inlined SVG, fixed colours
Files in the repository	none	one SVG per diagram
Needs a browser	on every build, and in CI	once, when a diagram changes
Covered by `format-check`	no	yes
Silent failure modes	two	none

Take this one if the project is Quarto-only, the diagrams are decorative rather than something a reader will select text out of, and a browser in the build is acceptable. Take the other if a diagram has to appear everywhere the prose does.