

The `mermaid` package (`mermaid.sty`)

Ryoya Ando
<https://ryoya9826.github.io/>

April 16, 2026

Abstract

`mermaid.sty` invokes the Mermaid CLI via `shellesc` and `\write18`, renders Mermaid diagrams during a $\text{\LaTeX} 2_{\epsilon}$ run, and embeds them as PDF. It can be used with engines such as `up \LaTeX` , `pdf \LaTeX` , and `Lua \LaTeX` . Rendering Mermaid diagrams requires the Mermaid CLI (`mmdc`, etc.) and headless Chromium (Puppeteer). Compile with shell escape enabled (`-shell-escape`).

1 Operating conditions

| Item | Condition |
|---------------|---|
| Format | $\text{\LaTeX} 2_{\epsilon}$ |
| Shell escape | <code>-shell-escape</code> (invoke the CLI via <code>shellesc</code> / <code>\write 18</code>) |
| External tool | Mermaid CLI (<code>mmdc</code>) |

If the Mermaid CLI is not present in your environment, `mermaid.sty` does not auto-install `mmdc` or similar. A typical Mermaid CLI installation uses **Node.js** with `npm` (or `npm`). If you set the package option `Renderer` to a command that includes `npm -y`, for example `npm -y @mermaid-js/mermaid-cli ...`, then `npm` may fetch and install packages as needed (**network access** may be required).

When compiling, unless the `Renderer` option names another command, the default renderer `mmdc` must be visible in the same environment (if you set `Renderer`, that command must be visible instead). The Mermaid CLI depends on headless Chromium (Puppeteer); see the Mermaid CLI documentation for details.

Log output uses `\typeout` with the `[mermaid]` prefix. If something goes wrong, check the `.err` files under `mermaid/`.

2 License and source

L^AT_EX Project Public License (LPPL) 1.3c or later.

Source and issues: <https://github.com/ryoya9826/ltMermaid>

3 Usage

3.1 Minimal document

```
1 \documentclass{article}
2 \usepackage{mermaid}
3
4 \begin{document}
5 \begin{mermaid}
6 flowchart LR
7   A --> B
8 \end{mermaid}
9 \end{document}
```

Compile example:

```
pdflatex -shell-escape yourfile.tex
lualatex -shell-escape yourfile.tex
uplatex -shell-escape yourfile.tex && dvipdfmx yourfile.dvi
```

3.2 Layout adjustments (optional)

By default, graphics included via `adjustbox` are scaled. Example:

```
1 \MermaidAdjustBoxOpts{max width=0.8\linewidth,center}
2 \MermaidAdjustBoxOpts{max width=0.9\linewidth,center,valign=T}
```

3.3 Beamer

The package can also be used with the `beamer` class. Every frame that contains a `mermaid` environment must use the `fragile` option (the environment depends on `fancyvrb`). Example: `\begin{frame}[fragile]{Figure}`.

```
1 \documentclass{beamer}
2 \usepackage{mermaid}
3
4 \begin{document}
5 \begin{frame}[fragile]{Mermaid}
6 \begin{mermaid}
7 flowchart LR
8   A --> B
9 \end{mermaid}
10 \end{frame}
11 \end{document}
```

4 Package options (optional)

- **Renderer:** Prefix for the renderer command. If omitted, `mmdc` is the default. You can specify it explicitly, for example `npx -y @mermaid-js/mermaid-cli`.

5 User commands

- `\MermaidRendererOptions{...}`: Additional CLI arguments before `-i / -o` (when PDF fit is enabled, merged after the built-in `-f`).
- `\MermaidNoPdfFit`: Disables `-f / --pdfFit` for `mmdc` (by default this is *enabled*).
- `\MermaidAdjustBoxOpts{...}`: Full list of `adjustbox` keys around `\includegraphics` (default: `max width=0.9\linewidth,center`).
- `\MermaidGraphicsOpts{...}`: Additional keys for `\includegraphics` (rotation, `trim`, etc.). Width is usually set with `\MermaidAdjustBoxOpts`.

6 Output files

Intermediate `.mmd` files and rendered `.pdf` files are written under the `mermaid/` directory in the directory from which you run the compiler. File names are of the form `mermaid-sample-mermaid-1.mmd`, `mermaid-sample-mermaid-1.pdf`, ..., using `mermaid-sample` (the job name) and a running index per diagram.

7 Diagram examples

Left: source typed into the `mermaid` environment. **Right:** rendered result.

```

1 \begin{mermaid}
2 flowchart TB
3   subgraph client["Client tier"]
4     WEB["Browser / SPA"]
5     CLI["CLI / batch"]
6   end
7   subgraph edge["Edge"]
8     GW["API Gateway"]
9   end
10  subgraph svc["Service tier"]
11    AUTH["Auth"]
12    API["Business API"]
13    WORK["Workers"]
14  end
15  subgraph store["Data"]
16    DB["PostgreSQL"]
17    CACHE["Redis"]
18    QUEUE["Job queue"]
19  end
20  WEB --> GW
21  CLI --> GW
22  GW --> AUTH
23  GW --> API
24  API --> WORK
25  API --> DB
26  API --> CACHE
27  WORK --> QUEUE
28  WORK --> DB
29 \end{mermaid}

```

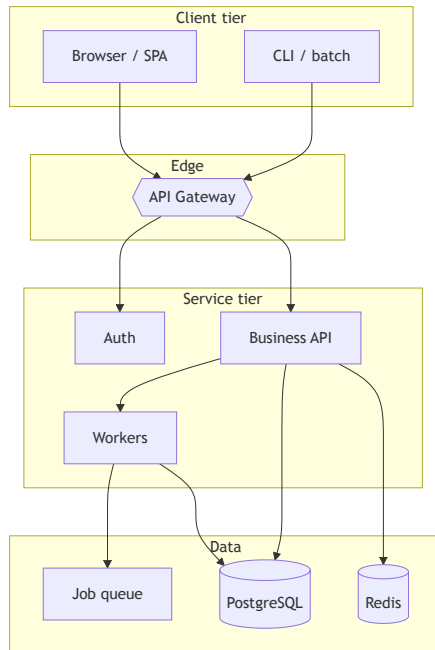


Figure 1: Multi-layer architecture (subgraphs and node shapes)

```

1 \begin{mermaid}
2 sequenceDiagram
3   autonumber
4   actor U as User
5   participant B as Browser
6   participant A as Auth API
7   participant S as Business API
8   participant D as DB
9   U->>B: Log in
10  B->>A: POST /token
11  A->>D: Verify user
12  D-->>A: Row
13  A-->>B: JWT
14  B->>S: GET /orders ( Bearer)
15  S->>A: Validate token
16  A-->>S: Claims
17  S->>D: SELECT
18  D-->>S: Rows
19  S-->>B: 200 JSON
20  B-->>U: List view
21 \end{mermaid}

```

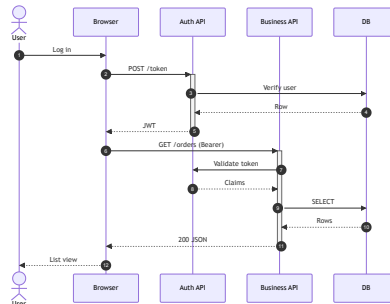


Figure 2: Sequence diagram (numbering, asynchronous arrows, multiple participants)

```

1 \begin{mermaid}
2 stateDiagram-v2
3   [*] --> Draft: Create
4   Draft --> Review: Submit
5   Review --> Draft: Send
6     back
7   Review --> Approved:
8     Approve
9   Approved --> Published:
10    Publish
11   Published --> Archived:
12    End
13   Review --> Rejected:
14     Reject
15   Rejected --> [*]
16   Archived --> [*]
17 \end{mermaid}

```

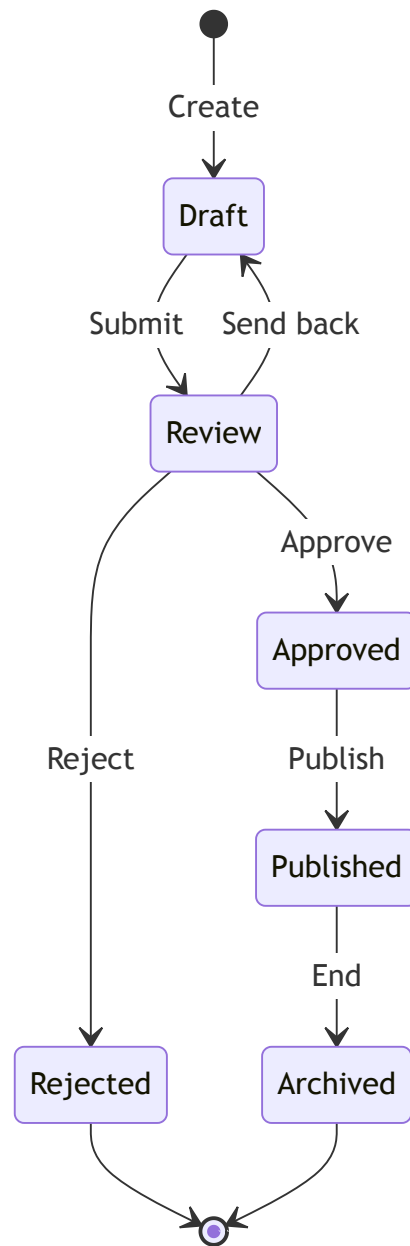


Figure 3: State transitions (`stateDiagram-v2`)

8 Revision history

Version 1.0 (2026-04-16): mermaid.sty stable release.

Version 0.1 (2026-04-15): Initial release (superseded by v1.0).