

The folder structure

ar013 · 7 July 2026

A demolab repo has a place for everything, and the layout is the same in every repo. Once you know the four content directories, the engine, and the root files, you can find anything by name. A fresh lab is born from `uvx demolab-cli init` in an empty folder, which lays down the whole tree below — everything in it is yours.

```
demolab/
├─ tools/           the science - one directory per tool (tool.py + tests)
├─ experiments/     the runners - one expNNN.py per experiment
├─ writings/        the writeups - one .typ per entry, by id
├─ artifacts/       the committed record of every run
│  ├─ data/<id>/    figures + numbers.json - the neutral record
│  ├─ pdfs/         compiled PDFs (shareable)
│  └─ site/         the built web site - GITIGNORED
├─ demolab.yaml     branding + collections - marks the lab root
├─ HOUSESTYLE.local.md optional house-style overrides
├─ .demolab/        staged engine assets - GITIGNORED
├─ AGENTS.md · CLAUDE.md · README.md · pyproject.toml · .github/
└─ temp/           short-lived run scratch - GITIGNORED
```

The four content directories

These are yours. `tools/` holds the science, one directory per tool, each with its CLI in `tool.py` and a test beside it. `experiments/` holds the runners: one `expNNN.py` per experiment, which invokes a tool, renders figures, and stages its output. `writings/` holds the writeups, one `.typ` file per entry: an `expNNN.typ` reads an experiment's run, an `arNNN.typ` is a prose-only article, and an `arNNN.slide.typ` compiles to a standalone slide deck. `artifacts/` is the committed record: `data/<id>/` keeps each run's figures and `numbers.json`, and `pdfs/` keeps the shareable PDFs. Both of those are in git; `artifacts/site/` and `temp/` are gitignored because the build regenerates them.

What ties an experiment together is its *id*, not a folder. The same `exp000` threads through all three: `experiments/exp000.py` runs it, `artifacts/data/exp000/` records it, `writings/exp000.typ` writes it up.

The engine and the root files

The engine is the *black box*. It holds the Typst publisher, the scaffold templates, the agent runbooks, and the guides, and it lives inside the installed `demolab-cli` package rather than in your tree, so you never hand-edit it. The CLI stages the few files Typst needs to read — `lib.typ` and the web assets — into the gitignored `.demolab/` directory at the lab root, and

refreshes it automatically. Updating demolab is an ordinary dependency bump: `uv lock --upgrade-package demolab-cli && uv sync`.

Two root files carry your configuration: `demolab.yaml` for branding and collections (it also marks the lab root), and the optional `HOUSESTYLE.local.md` for house-style overrides. The remaining root files, `AGENTS.md`, `CLAUDE.md`, `README.md`, `pyproject.toml`, and the CI workflow, are thin stubs laid down once by `init` — yours to edit, and never touched by an update.

Learning it interactively

Type **STRUCTURE** in capitals and the coding agent will walk you through this tree, path by path, in your own repo.

The full reference lives in `STRUCTURE.md`.