nixss†: a static site library for Nix

codeberg.org/xlambein/nixss

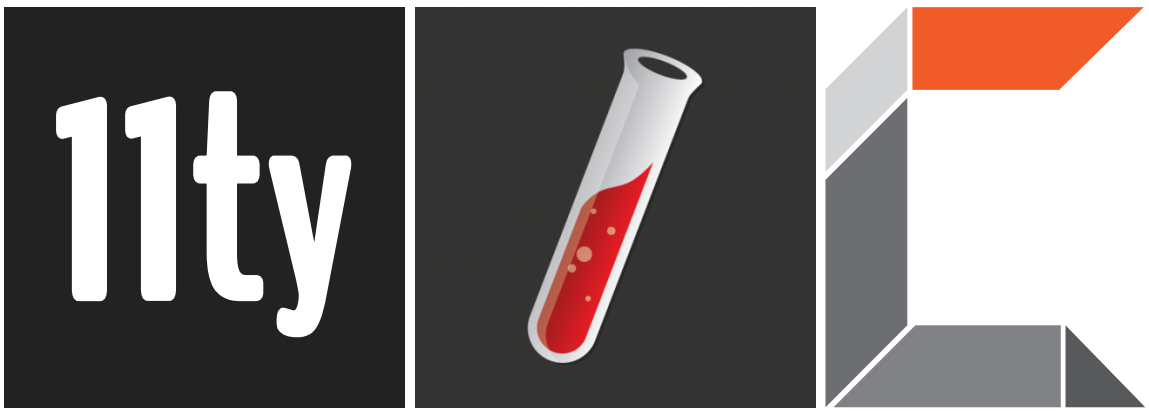
† pronounced “nix” but you're a snake

Xavier Lambein

www.lambein.xyz

@xavier@sunny.garden

Making static sites the normal way



What if you need more control?

Roll your own!

Static site generator =
file conversion + build system

make and pandoc

Finally, a static site generator for the 1990s

```
all: output/index.html output/about.html

output/%.html: pages/%.md
    pandoc --self-contained $^ -o $@

clean:
    rm -r output/*
```

Pandoc is extremely powerful! This might be enough for your usecase.

But we can do better

A Nix library for building static sites

```
{nixss}: let

  # Processors tranform pages
  md2Html = nixss.pandoc {
    from = "markdown";
    to = "html";
    # ...other pandoc(1) command-line arguments
  } (nixss.replaceExt "md" "html");


  # Apply processor over each file in directory
  pages = nixss.mapDirectory md2Html ./pages;

in

  # Pages are aggregated to make a site
  nixss.directory {
    filename = "www";
    src = pages;
  }
```

Attributes of `nixss` pages

```
page = (derivation {...}) // {

  # Filename of the page, e.g. `index.html`
  filename = "index.html";

  # Text content of the page, computed lazily if possible
  text = "...";

  # Source of the page (if it exists), either a single page or a list of pages
  src = «derivation»;

  # Metadata about this page
  metadata = {
    # For example:
    title = "My website!";
    pubdate = "2026-01-32";
    # ...etc
  };

}
```

Example: generating an index

```
{nixss, lib}:
let

  # The pages of our site
  pages = nixss.mapDirectory myProcessor ./pages;

  # Build a list of HTML links from the page titles and filenames
  list = lib.concatMapStrings
    (p: ''<li><a href="./${p.filename}">${p.metadata.title}</a></li>'')
    pages;

  # Concat these links into an HTML list
  index = nixss.text {
    filename = "index.html";
    text = ''<ul>${list}</ul>'';
  };

in

  nixss.directory {
    filename = "www";
    src = pages ++ [index];
  }
```

Nix as a templating language

```
let
  template = {
    title ? null, # From `page.metadata.title`
    content, # From `page.text`
  }: ''
    <!doctype html>
    <html lang="en">

    <head>
      <meta charset="utf-8">
      <title>${
        if title == null then "" else "${title} &mdash;"
      }My website</title>
    </head>

    <body>${content}</body>

    </html>
  '';
in
  nixss.template.instantiate template {} some-other-page
```

The `nixt` "templating language"

Let's abuse `with` and import-from-derivation:

```
let
  instantiate = page: let
    env = { inherit (page) filename; } // page.metadata;
    text = "__env: with __env; ''${page.text}''";
  in
    import (builtins.toFile "crimes.nix" text) env;
in
  instantiate ./index.md.nxt
```

Now we can use Nix like it's PHP!

```
# Welcome to my website!

This is a **Markdown page**...

${foreach page (page: with page; ''
- [${metadata.title}](./${filename})
'')}

...or is it a _Nix string_?
```

codeberg.org/xlambein/nixss

Xavier Lambein

www.lambein.xyz

@xavier@sunny.garden