Software design and publishing manifesto for darescience.net

Mark Rickerby, May 1 2021

Objective

Publish a group blog for the DaRe Science research collective, to go live in May or June 2021.

This document should also function as a short publishing manifesto/tutorial to get you started with your own personal websites or special topic blogs and newsletters.

This is all stuff you can learn if you're interested, or you can stick with just adding Markdown posts and images to the content folder.

Components

- NodeJS and Eleventy
- Content graph
- Styleguide and template theme
- Contributor agreements and editorial workflow

NodeJS and Eleventy

Justification

Almost any programming language can be used to generate a small blog or similar content website. Most of the content specs provided here can translate across different software without major changes.

The choice of JavaScript via NodeJS is mainly to keep things simple to run in multiple environments and transfer between different automated CI and publishing services. Because of the prevalance of JavaScript build tools for many common web tasks in 2021 (embedding charts and data viz, generating CSS, compressing images, generating snapshots for social media images, etc), many site generators written in other languages end up having JavaScript/NPM as a requirement anyway. Given this situation, it makes sense to just install one thing via one platform (NPM), rather than go through the motions of installing a whole chain of cross-platform dependencies.

Eleventy isn't perfect but it currently has the best setup and publishing experience in JavaScript land and is mostly painless once you write a good set of templates.

For data science/engineering/design authors, if anyone is going to be running it locally, I strongly advise against installing larger more complex and bloated JavaScript/React generators like Gatsby. Fine for web designers, but it will get messy fast if you just want to write and publish stuff. If you do get to the point of wanting explorable, interactive essays come and talk to me again on that and we can look at the skills and strategies needed for that to happen.

Alternatives

Ruby would make a good alternative if you want to use any of the custom graph query language systems I'm working on for managing static sites and interactive fiction projects but this might require a bit more data wrangling than only designing a template theme. Probably best to wait until I've released my research paper on this later in 2021, rather than jump in right now.

We could also use Hugo or a Python setup if that is what authors are more comfortable with but there may be more template complications here as these tools are not as streamlined as Eleventy for basic web tasks.

Technical Requirements

- Shared Git repository (with a private or public web origin)
- NodeJS and NPM installed for everyone who wants to build the site locally
- Origin CI/build server web service (eg: Netlify) if you want to do cloud rebuilds on push
- Editing tools able to open Markdown and HTML files and edit web images
- Git installed for all contributors

Content Graph

Could also be called 'Content Directory' or 'Content Manifest' but I prefer thinking of it in terms of graphs as working with content at the level of its data structure rather than its presence on the filesystem is more suitable for thinking through editorial design and templates.

Filesystem Structure

All authored files and documents go into a hierarchy underneath a central root directory, usually called `content` or `docs`.

Option A)

```
/content/posts
/content/profiles
/content/pictures
/content/about.md
```

Option B)

```
/content/articles
/content/authors
/content/images
/content/about.md
```

Option C)

Any similar naming variations that work best for your preferences.

Decisions

If the site stays roughly the same for a long time, it doesn't make much difference what names you choose. It may matter if you want to add more content types to the site in future.

When the static site generator runs, this is parsed off the filesystem and resolved to a memory graph with two collections of content nodes, eg: `posts` and `profiles` / `articles` and `authors`.

From this same structure, a whole lot of generated content patterns become possible, so it's very easy to change templates around or add new features without affecting any of the existing content.

Front Matter Dictionary

All popular static site generators support extending Markdown with a block of ‘front matter’ in the leading section of the document, providing a dictionary of key-value attributes (usually in YAML, although JSON and TOML are also sometimes used). Some static site generators impose a particular structure or schema for this metadata but others are open-ended and allow anything to go there.

Many more specific data types will go in and out of YAML and dynamically typed hashes as strings but some editors and generators do support specific data types in Markdown front matter (mostly dates).

Identfier

A short alphanumeric dash-separated text identifier that can be used as a unique ID and a readable path segment in URLs, biography pages, category references, etc.

Examples:

```
gvdr
giulio-dallariva
the-title-of-a-blog-post
```

Text Line

A line of plain text with no leading or trailing whitespace/control characters. No HTML or embedded syntax unless supported by a specific site generator. Can contain punctuation and unicode characters

Examples:

```
The title of a blog post
```

ISO-8601 Date

Timestamps should be stored in the ISO-8601 date format. The advantage of this format is that it allows for trailing components of the timestamp to be optional. For instance, the following Mayday dates should all be valid (although they might produce different results if there is more than one entry published on the same day):

```
2021-05-01T16:00:00+12:00
2021-05-01
```

This is sometimes useful when hand-editing posts as you shouldn’t have to care about specifying the exact minute and hour when a post went live, just the day you want to publish it on.

The following dates are also valid although not recommended.

```
2021-05
2021-05-01T02:00:00Z
```

If you enjoy using command line scaffolding, you could write your own tool for creating draft posts from a template—this will automatically set date fields to the current timestamp and add any other text blocks you need so you’re not starting from scratch each time (though in practice, most people just copy/paste one of the existing `.md` files and go from there—this is the great thing about static site generators, it allows you to work however you want).

URI

URI fields should be either a full URL with DNS info or a local URI path reference.

```
https://javascriptcdn.net/packages/graph-algorithms-lib/1.0.1/index.js
/images/cards/project-banner-img.png
```

Twitter Summary Card

Allowed values are:

```
summary
summary_large_image
```

This can be completely ignored most of the time The templates can autogenerate a card embed—it just won't be as sharable and impacful as as embedding a custom image for the post. How to handle social sharing mostly depends on the intended audience and how much time you have to polish and edit particular posts before publishing them and moving on.

Blog Posts

Each blog post should be written as a single Markdown document (`.md`) that includes all the main body text and relevant metadata associated with the piece.

Metadata

The core post metadata requirements for a science/research blog are:

- Basic publication information (title, summary, dates, etc). If not imposed by the static site generator, my recommendation is to echo the format of the Atom syndication spec so there is always a 1:1 translation between the written markup and a generated RSS/Atom feed.
- Social media embed properties and microdata. Various social media platforms and search engines will expand linked URLs into an embedded card display. Some of the basic publication details can be substituted here but there are some properties unique to social sharing which need to be handled explicitly (as well as trying to normalise the inconsistencies between different platforms).

Optional extras:

- Categories, topics, tags, related content and shared vocabularies. Used for organising listings of specific themed posts and making them findable elsewhere in the right context. This is site-specific and optional. It doesn't have to be decided at the outset and can be layered on in future once a better idea of recurring areas of interest and themes emerges. This isn't so valuable when the site contains a small number of posts and does require some effort and coordination between authors to maintain a consistent set of keywords over time.

You can also add extra fields like `abstract` and `references` to the posts, but if this starts to get more complicated, it's probably better to arrange the site in collections of different document types (eg: Research Projects, Papers, Blog Posts).

Post URLs will be generated based on the filename of the document.

YAML Schema

All fields are optional except for those marked required.

```
---
title: <Text Line> (required)
date: <ISO-8601 Date> (required)
summary: <Text Line>
subtitle: <Text Line>
author: <Identifier>
series: <Identifier>
meta:
  title: <Text Line>
  description: <Text Line>
  image: <URI>
  twitter_card: <Twitter Summary Card>
```

The `author` field can also be replaced with a plural `authors` list, supporting blog posts or articles written collaboratively by a group of people:

```
authors:
  - <Identifier>
  - <Identifier>
```

Identifiers specified in `author` or `authors` should map to author profiles, as specified below.

The `series` field is optional but could be used to link multiple posts together in a series with multiple parts.

Example Post

  ---
  title: This is my post
  date: 2021-05-01
  summary: An exciting summary of why you want to read it
  author: maetl

  A first paragraph here with some *formatted text*.

  ```julia
  # An embedded code block with syntax highlighting
  \```

  A second paragraph here with [a hyperlink](/go/somewhere/else).
```

Author Profiles

Each author profile should be written as a single Markdown document (`.md`) with metadata associated with the author in front matter and a short bio as the main page content.

YAML Schema

```
---
id: <Identifier> (required)
name: <Text Line> (required)
website: <URI>
```

Example Profile

```
---
id: maetl
name: Mark Rickerby
website: https://maetl.net
---

Mark studies software architecture and narrative systems with a particular
interest in using graph databases and graph modelling to improve editorial design
workflows for large scale media and storytelling projects (including videogames,
films, websites and novels).
```

Styleguide and template theme

This encompasses the majority of undiscovered design work that needs to be done to get the site looking good.

Styleguide Elements

- Logo
- Colour palette
- Typography specs
- Layout specs

Template Sitemap Sketch

Core Templates

An article/post template an archive listing, and a homepage or about section is the bare minimum required to get a site working:

- Homepage/About
- Article/Post
- Archive/Listing

The choice of templates and names may impact on URL structure but for the most part, the content names will pass through directly from source to output site.

eg:

```
/content/home.md          -->  https://darescience.net/

/content/about.md         -->  https://darescience.net/about/

/content/posts/my-post.md  -->  https://darescience.net/posts/my-post/

/content/posts/*          -->  https://darescience.net/posts/
```

Profile Option A)

A separate profile URL for each author:

Eg:

```
/content/authors/giulio-dallariva.md
```

```
                    -->  https://darescience.net/authors/giulio-dallariva/

/content/authors/mark-rickerby.md
                    -->  https://darescience.net/authors/mark-rickerby/
```

Profile Option B)

Single URL for the whole site with profiles listed inline and linked with `#id`s within the document:

```
/content/authors/giulio-dallariva.md
                           -->  https://darescience.net/authors/#giulio-dallariva

/content/authors/mark-rickerby.md
                           -->  https://darescience.net/authors/#mark-rickerby
```

The main source of inbound links to the profile pages will be from the author byline on articles/posts.

Other Templates Needed

Not definitive yet and the site can probably do without more complex listings but for thoroughness:

- Category/Topic pages
- A-Z Index
- Impressum/Colophon/Contact/Code of Conduct/Acknowledgement of Country pages

Contributor agreements and editorial workflow

If the site has multiple authors it can either have a copyright statement that declares ownership rights or a Creative Commons license declaring how the site should be credited when remixed and shared.

An editorial workflow should be agreed beyond the constraints of Git commits and merging branches. This should be written down so the decision is clear at the outset and new people can be brought onto the team more quickly in future with a process in place for accepting their contributions.

Questions and design prompts to think about:

- Is the site going to build from a branch on a CI server or be pushed live manually by an editor on the team?
- Is there a signoff/peer review process before an article goes live on the web or would it work better to publish rough drafts publicly, then collaborate on feedback and critique to improve it for a few days after publishing?
- Is there a shared doc with a vision for the collective and its publishing aims? Could these ideas be worked into editorial guidelines or a list of ideas for future topics to write about?
- What sorts of styleguide decisions and creative tools would help new authors get started more easily?