Tools and Workflows for Collaborating on Static Website Projects

Static website generators have seen a significant increase in popularity in recent years, offering many advantages over their dynamic counterparts. While these generators were typically used for blogs, they have grown in usage for other web-based projects, including documentation, conference websites, and image collections. However, because of their technical complexity, these tools can be inaccessible to content creators depending on their level of technical skill and comfort with web development technologies. Drawing from experience with a collaborative static website project, this article will provide an overview of static website generators, review different tools available for managing content, and explore workflows and best practices for collaborating with teams on static website projects.

By Kaitlin Newson

Introduction

In 2017, the Ontario Council of University Libraries’ Geo Community wrapped up a 2-year digitization project of over 1,000 historical topographic maps of the Ontario region. As part of this project, the team wanted to create an online presence for these maps to highlight this collection, promote the project, and make the maps more accessible to a wide audience of users. After evaluating different web technology options, the team decided to use a static website generator to create this online project space.

Static Site Generators

Static website generators are tools used to generate a website made up only of HTML, CSS, and JavaScript. Static websites, unlike dynamic sites, do not use databases or server-side scripting languages, and the website appears to the user as it exists on the web server. A static site generator is typically made up of:

- A templating language for website layout and theming.
- A markup language for content creation (e.g. Markdown).
- A local development server to preview and test the site before building.
- A compile process that builds the final site files into HTML, CSS, and JavaScript (Rinaldi 2015).

To those that have been working in web development for a while, the re-emergence of static websites may seem like a step backwards. In the beginning of the web, websites were all static, made up only of HTML, and eventually adding CSS and JavaScript. As web technologies developed, the limitations of static websites became apparent, and databases and server-side scripting languages began to fill the gaps in functionality left by the static web. With the emergence of large-scale content management systems (CMS) like WordPress and Drupal, dynamic websites became more popular as plugins and editing systems led to tools that could be used by content creators without the need for a depth of technical expertise.

So, why has the shift back to static websites emerged? With the creation of the modern static site generator, static sites no longer need to be hand-coded from scratch. Getting a static website live can be a straightforward process for those with the technical skill to create them. With services like GitHub pages, there’s no need to worry about managing a server or a domain, although a custom domain can still be used – in fact, that is how code4lib’s own conference website is developed, using the Jekyll static site generator. For many websites, the process of installing a CMS, managing a database, and keeping the site up to date to protect against security vulnerabilities can be cumbersome and overly complex. Static website generators offer a more straightforward approach, with pre-existing themes that can easily be used to create a polished and modern website with minimal need for maintenance. There are also many to choose from, with almost 200 options written in a variety of programming languages currently available according to StaticGen.

Features

While static site generators are not suitable for every website, they have a number of benefits that should be considered when evaluating systems to use for a new web project. Static websites can be up to six times faster than their dynamic counterparts, primarily due to the challenges of
caching dynamic content which are not a concern in a static environment (Christensen 2015). These websites are also easier to secure since there is no database to be exploited or outdated plugins to create vulnerabilities.

From a preservation perspective, static websites are much easier to preserve. Since dynamic websites have content pulled from a database and can generate different views based on the user’s context, they can present more challenges for archiving, especially without direct access to the website's back-end. In the case of a static website, preservation is much easier because the site is displayed as it exists on the web server (Rumianek 2013).

Hosting is also much simpler with static websites. Services like GitHub pages have made hosting much more straightforward, with no need to manage a server or even have a registered domain. If you are using your own server, a static site means that you don’t need to worry about managing a database or working with server-side programming languages.

However, static websites won’t meet the requirements of every web project. Having no database or server-side scripting means that some commonly used website functionality like user accounts or input are more challenging, if not impossible, to implement. Many static website generators have plugin options, but many of these plugins require integration with third-party services that may not be ideal depending on a project’s privacy requirements. For example, if you want to have a contact form on a static website, using a service like Formspree is required, meaning that all of your data is passed through this service and the functionality of this feature depends on the reliability of this service.

Another challenge with static websites is their lack of a straightforward administrative environment like those found in CMSs such as WordPress or Drupal. A lack of editing environment can be a major barrier for content creators, who may not be comfortable writing content in Markdown, using command line tools, or managing a git repository. However, solutions are gradually being developed to tackle this problem.

Content Management Tools

One of the major disadvantages to static website generators is their lack of a content authoring environment, which can be a significant challenge for content creators that are not as familiar with web technologies. When working as a team on a project, this limitation may result in a higher workload for team members with web development expertise, and can ultimately slow down the project's progress. Since content creation for websites is frequently a collaborative process, there is a clear need to have user-friendly systems that allow users to edit content regardless of their level of technical skill or knowledge of web development.

Since static website content is written in Markdown, many Markdown editors, such as StackEdit, can be excellent options for creating website content. Unfortunately, many of these editors do not have the same level of integration with Github as other tools or are missing important features that content creators may need, such as image import or interface elements for formatting text. Markdown editors can be a valuable tool for content creators, but most do not go far enough to create a fully integrated environment that is accessible to users of different technical skill levels.

As static website generators have grown in popularity, developers have begun to address the gap in editing environments by creating more robust CMSs that can be used with static websites, with 27 options currently listed at headlesscms.org. While many of these services are paid, some are fully open-source and can be configured by a website administrator to be used by other project collaborators. NetlifyCMS is emerging as a strong contender for a static site CMS to bridge the gap between raw Markdown editing and the WordPress-like editing environment that many are currently used to (fig. 1). This tool allows users to log into the administrative interface with their GitHub credentials, manage different types of content, and write using a minimal and user-friendly interface.

![Figure 1. The Netlify CMS, built from the demo available at https://www.netlifycms.org/docs/test-drive/](https://www.netlifycms.org/docs/test-drive/)

If a static website project is using GitHub, one option is to use the built-in editor for content editing on their website (fig. 2); however, this does not give users straightforward options for text formatting, and a working knowledge of the markup language being used (e.g. Markdown) is necessary. This approach is also missing functionality that content authors may need, like the ability to import images. While Markdown is not a challenging syntax to learn, it’s important to make the content creation as simple as possible for users with a range of technical skill levels. By using more robust content management systems, users have more editing options available to them without requiring technical knowledge of web development.
Figure 2. The GitHub editing environment

Prose (fig. 3) works with an existing public GitHub repository to allow users to author content in a clean and simple environment. Once editing is complete, content can be committed directly to the GitHub repository, meaning that no working knowledge of git is required. Prose has many standard text editing features, like font bolding and inserting headings, as well as the option to import images that will be inserted directly into the repository. Prose also has minimal setup requirements, and only needs GitHub authorization to access the project's repository to get started with content management.

Figure 3. The Prose editing environment

It should be noted that the tools above rely on a project using a GitHub repository. Internal repository management tools like GitLab or Bitbucket have a built-in web editor similar to that of GitHub but do not integrate with the tools mentioned above without some code configuration. Desktop editors for Markdown writing may meet content editing needs in these cases but were not explored in depth for this article.

The Topographic Maps Project

In 2017, members of the Ontario Historical Topographic Maps Digitization Project team developed their website using a static website generator. The Historical Topographic Maps Digitization Project was a 2.5-year digitization initiative completed by the geography community in the Ontario Council of University Libraries (OCUL). The project involved the digitization and georeferencing of over 1,000 topographic maps across Ontario. One of the final steps of the project was promotion, which involved the development of a website to showcase the project and the digitized maps. Since the full resolution maps and metadata were being stored and displayed through Scholars GeoPortal, this website was intended to promote the project and provide a user-friendly way to browse the collection, rather than to provide access to geospatial mapping tools, preservation capacity, or complete metadata records.
The Historical Topographic Map Digitization Project was initiated by the Ontario Council of University Libraries (OCUL) GeoCommunity in 2014. It is a 2.5 year initiative to digitize and georeference early-twentieth century historical topographic maps covering the province of Ontario at the scales of 1:25,000 and 1:63,360. The goal is to create and provide access to high-resolution map scans that preserve historical topographic information and meet the needs of current and future users.

The project team was made up of staff from across OCUL universities and Scholars Portal. As the website lead, I had prior experience working with web technologies, including HTML, CSS, JavaScript, git, and content management systems. This prior experience gave me the necessary knowledge to handle the technical aspects of the project, while other team members focused on content.

Initially, we were unsure of what tool we wanted to use for the project’s website. We began by considering content management systems like Drupal and Omeka, but these seemed more complex than necessary for a website primarily made up of images and text. Scholars Portal had previously used a static site generator for another project, so I began investigating different functionality and theming options to determine if this was suitable for the new website. After creating a demonstration site and sharing it with the rest of the project team, we decided to develop the site using a static generator.

Hugo, a popular static generator built in the Go language, was used to create the project website. Hugo was selected because of the simple installation process, fast build speeds, and previous experience using the tool. The website is hosted on the Ontario Library Research Cloud (OLRC), a cloud storage service built with OpenStack Swift that has the ability to serve static content. This project serves the dual purpose of showcasing the collection and demonstrating a potential use case for institutions using the OLRC.

The theming for the website was based on the Creative Portfolio theme for Hugo, which was originally created for designers to showcase their work, but works well to highlight any collection of images. The template was modified to add multilingual support for English and French, along with some CSS modifications to improve the colour contrast and font sizes for accessibility purposes. JavaScript and jQuery were used for additional functionality, including magnifying the maps when a user hovers over them with their mouse (fig. 5), and filtering the list of maps based on text input. An inventory of the full collection of maps had been previously created as a Google spreadsheet, so this data was converted to JSON which was then used to generate the listing of the collection. A Google App Script was used to export the spreadsheet data into usable JSON (Fox 2013). JuxtaposeJS was used to create a comparison slider to demonstrate changes between map areas over time (fig. 6). The code for the website can be found in the project’s GitHub repository.

Figure 4. The Historical Topographic Maps Digitization Project website, available at ocul.on.ca/topomaps

The project team was made up of staff from across OCUL universities and Scholars Portal. As the website lead, I had prior experience working with web technologies, including HTML, CSS, JavaScript, git, and content management systems. This prior experience gave me the necessary knowledge to handle the technical aspects of the project, while other team members focused on content.

Initially, we were unsure of what tool we wanted to use for the project’s website. We began by considering content management systems like Drupal and Omeka, but these seemed more complex than necessary for a website primarily made up of images and text. Scholars Portal had previously used a static site generator for another project, so I began investigating different functionality and theming options to determine if this was suitable for the new website. After creating a demonstration site and sharing it with the rest of the project team, we decided to develop the site using a static generator.

Hugo, a popular static generator built in the Go language, was used to create the project website. Hugo was selected because of the simple installation process, fast build speeds, and previous experience using the tool. The website is hosted on the Ontario Library Research Cloud (OLRC), a cloud storage service built with OpenStack Swift that has the ability to serve static content. This project serves the dual purpose of showcasing the collection and demonstrating a potential use case for institutions using the OLRC.

The theming for the website was based on the Creative Portfolio theme for Hugo, which was originally created for designers to showcase their work, but works well to highlight any collection of images. The template was modified to add multilingual support for English and French, along with some CSS modifications to improve the colour contrast and font sizes for accessibility purposes. JavaScript and jQuery were used for additional functionality, including magnifying the maps when a user hovers over them with their mouse (fig. 5), and filtering the list of maps based on text input. An inventory of the full collection of maps had been previously created as a Google spreadsheet, so this data was converted to JSON which was then used to generate the listing of the collection. A Google App Script was used to export the spreadsheet data into usable JSON (Fox 2013). JuxtaposeJS was used to create a comparison slider to demonstrate changes between map areas over time (fig. 6). The code for the website can be found in the project’s GitHub repository.
The initial draft for website content was created collaboratively in Google Docs, with different team members contributing individual sections. Through team meetings, decisions were made about different aspects of the website, such as top-level navigation items, content structure, and the display of the maps. Once the team agreed that the content was complete, it was sent for French translation. Members of the project team then used Prose to work with content in the GitHub repository. As content was developed, some bugs were encountered when editing with Prose which prevented changes from being committed to the repository, and the built-in GitHub editor was used for content modifications in these cases. When changes were committed to the repository, the website lead would then pull down the changes, recompile the site, and update the development version of the site.

Once all of the content was in place, the team did a final review, referring any styling suggestions to the website lead, and fixing any content issues that were found. Final configuration elements were added, including Google Analytics tracking, and the site was launched and promoted to the academic community.

Overall, the Historical Topographic Maps Project website was a success for the OCUL Geo Community. The website serves as an attractive and intuitive way for users to engage with the maps that may not have been exposed to them otherwise. While more advanced geospatial data users may explore the maps in their preferred GIS tool or through Scholars GeoPortal, the website provides a space for a broader range of users to engage with the map collection.

Conclusions & Lessons Learned

For the team at Scholars Portal, this project helped us to better understand what kinds of projects static generators are best suited for, and how they can be used collaboratively. Although we do have sites built with Drupal, these large content management systems are far too resource-intensive and complex for small and simple websites. Static website generators still require technical knowledge to get them up and running but require far less long-term maintenance than something built in Drupal or WordPress. Once you understand how to use them, these tools can be incredibly efficient for getting a simple and attractive website up quickly, without needing to manage a database or write server-side code; however, an understanding of markup languages and command line tools is necessary. If you want to do customizations for your site, you may also need to understand CSS and JavaScript, and also be willing to learn the templating language that is used by the static generator. Depending
on your website hosting environment, you may also need knowledge of web server management. This technical knowledge is currently necessary
to get a static site up and running, but hopefully future technologies will reduce this barrier to entry.

Static websites have seen a strong resurgence with the modern static generator, offering a number of benefits over dynamic websites in
performance and security. While the editing environment isn’t yet on par with those found in content management systems like WordPress, tools
are gradually emerging to fill this gap in the development process. Static generators are not suitable for every kind of website, such as those that
require dynamic elements like user input, but should be strongly considered as one possible option when evaluating systems to use for a new
project.

### Notes
- If you’re interested in making a site with Jekyll and GitHub Pages, Programming Historian has an excellent tutorial to get you started.

### References
- Christensen, MB. [updated 2015 Nov 02]. Why Static Site Generators Are The Next Big Thing [Internet]; [cited 2017 August 16]. Available from:
- Fox, P. Exporting a Google Spreadsheet as JSON [Internet]. [updated 2013 June 07]. [cited 2017 August 16]. Available from:
  http://blog.pamelafox.org/2013/06/exporting-google-spreadsheet-as-json.html
  http://www.dlib.org/dlib/january13/rumianek/01rumianek.html

### About the Author

Kaitlin Newson (kaitlin@scholarsportal.info) is the Digital Projects Librarian at Scholars Portal, the technology service arm of the Ontario Council of
University Libraries. You can find her online at www.kaitlinnewson.com and on twitter as @kaitlinnewson.

Subscribe to comments: For this article | For all articles

---

This work is licensed under a Creative Commons Attribution 3.0 United States License.