Lesson Plan

This lesson plan provides an assortment of learning modules for teaching static web technologies for digital scholarship and scholarly communications librarianship. Each topic includes a learning objective and recommended readings, viewings, or tutorials for use in workshops or seminars.

Introduction

Learning Objective: Participants will be able to recognize the strengths and limitations that static websites provide scholarly communications librarians.

Static websites can facilitate the distribution, access, and longevity of digital scholarship. A static website is simply a set of files that exist somewhere on a public web server. While they provide some benefits to librarians who facilitate the dissemination and preservation of the scholarly record, they –by themselves– are inadequate for satisfying the full range of scholarly communication technology needs.

Benefits for Digital Scholarship

- Static websites **support a variety of digital scholarship and scholarly communications use cases**, including scholarly monographs, academic journals, digital exhibits, reproducible research, open educational resources, data visualizations, and digital library projects.
- Static websites are **built on open source software and open web standards**. There's nothing proprietary to the underlying technologies.
- Static websites can be **hosted anywhere**, enabling the freedom to deploy to (and migrate from) any hosting provider.
- Static websites rely on **fewer pieces of technical infrastructure** than dynamic web platforms: no databases, no software updates, no server maintenance, no security patches
- Static websites are **cheaper to maintain and preserve** than websites run on content management systems

Challenges and Drawbacks

- Static websites are more difficult to learn *because* they don't typically include a content management system. Instead, they require direct interaction plain text files via a text editor and command line interface. Static websites *can* be used with content management systems; however, this usually requires additional configuration.
- They are more difficult to edit after the work is complete. For scholarly publications and digital scholarship projects, frequent editing and updating may not be a primary concern, but projects involving many content editors with on-the-fly content updates may be better served by a publishing platform with a content management system.
- They do not support "logged in" user experiences. Static websites are easiest to make when they are public and read-only. Administrative dashboards, analytics, submission management systems, and subscriber content would need to be handled using other services.
- This is not unique to static websites; however, by taking on more control over the semantic markup and styles by editing and developing templates, you assume more responsibility over the usability and accessibility of the website. Using "as-is," warranty-free open source software requires an added level of care over the quality of the code. For example, one cannot assume that a popular open source theme for a static site generator meets your institutions digital accessibility policies. It's your responsibility to test the website and make improvements accordingly.

Deciding to use a static website for a digital scholarship project requires a thorough understanding of the project's use cases. For example, while static websites may be excellent options for digital exhibits and scholarly web publications, they are not suitable alternatives for repository systems or pre-print servers.

Reading: Newson, Kaitlin. 2017. "Tools and Workflows for Collaborating on Static Website Projects." *The Code4Lib Journal*, no. 38 (October). .

Newson introduces static site generators for digital library projects with a case study involving a digitized maps collection. Newsom explains how static site generators work and discusses their advantages and disadvantages for team projects.

Static vs Dynamic Websites

Learning Objective: After examining this topic, participants will be able to describe the differences between static and dynamic websites.

Most of the websites we use every day are dynamic websites. Dynamic websites store content in databases and use web servers to generate web pages upon each visit. Often, dynamic websites are managed by a content management system (CMS), like WordPress, Drupal, or Omeka. This make them attractive for websites that require permissions-based access, e-commerce, or social interactions. However, dynamic websites need continuing resources for software updates, maintenance, and security.

Static websites do not use databases to store content or web servers to dynamically generate web pages; instead, the web pages are pre-built on a personal computer using a static site generator and stored as static HTML files on a public web server, thereby eliminating the need for database security and routine software updates for servers. This makes them cheaper to host and easier to maintain.

Static websites used to make up the majority of the web in the 1990's. People would write HTML code (i.e. "markup") for every page and upload the files to a public web server. Today, people use **static site generators** to automate and simplify the process for making websites. "Think of a static site generator as a script which takes in data, content and templates, processes them, and outputs a folder full of all the resultant pages and assets" (Hawksworth, 2020). Compared to dynamic website platforms, static site generators are relatively small pieces of open source software we can run on our computers ourselves.

Tutorial:

Williamson, Evan. 2020. "Introduction to Creating Websites with GitHub Pages and Jekyll." Go-Go Gh-Pages! *This is an excellent introduction to using GitHub for static site publishing. From the website: "This workshop will introduce using free hosting from GitHub Pages integrated with the popular static website generator Jekyll. Along the way we will cover the basics of GitHub, HTML, Markdown, and Jekyll. You will learn how to set up a project repository, write content in Markdown, and publish your site, all using GitHub's user friendly web interface. More advanced usage of Jekyll for local web development is introduced final section." Follow along with the videos for the full workshop experience.*

Static websites have gained in popularity in the open source web development community with the rise of JAMstack. JAMstack is an architecture for building websites using static site generators (of which there are hundreds of open source options). JAMstack separates the front-end interface from the backend database in order to produce websites that are less vulnerable to software degredation and security risks. When publishing scholarly texts online, scholarly communications librarians rarely need backend database features. We primarily need the front-end to provide stable, unlimited online access to web publications.

Frontend Features	Backend Features
Public HTML interface	Server-rendered web pages
Metadata (JSON/HTML)	Storing user credentials
Full-text PDF download	Managing e-commerce transactions

Viewing: Watch at least first 6 minutes of "What is the JAMstack? and let's BUILD one"

Follow-up activity: Examine three publishing tools from the Catalogue in "Mind the Gap: A Landscape Analysis of Open Source Publishing Tools and Platforms" and make a case for why each tool is or is not an example of JAMstack in scholarly communications.

Plain Text vs Rich Text

Learning Objective: Produce and transform plain-text documents for editing, publishing, and archiving

Static site generators require content and styles to be stored as *plain* text. There are two main types of documents we use to write and edit text: plain-text and rich text. Most of us are trained to use rich text editors: emails, word documents, content management systems. This is for good reason: they're easy to use and we need them for everyday things. Plain text exposes the raw, semantic characters within a document, whereas rich text displays the formatting features and styles. For librarians, plain text offers some advantages over rich text, as Tenen and Wythoff (2014) explain:

> Plain text both ensures transparency and answers the standards of long-term preservation. [Microsoft] Word may go the way of Word Perfect in the future, but plain text will always remain easy to read, catalog, mine, and transform. Furthermore, plain text enables easy and powerful versioning of the document, which is useful in collaboration and organizing drafts. Your plain text files will be accessible on cell phones, tablets, or, perhaps, on a low-powered terminal in some remote library. Plain text is backwards compatible and future-proof. Whatever software or hardware comes along next, it will be able to understand your plain text files.

	File Formats	Editors
Plain text	`.xml`, `.html`, `.md`	Notepad, TextEdit, Visual Studio Code
Rich text	`.docx`, `.rtf`, `.odt`	Microsoft Word, Scrivener

Coming to a plain text editor from a word processing program (like Microsoft Word), might feel like writing computer code rather than text for humans. That is because there is little material difference between plain text and code. Plain text is the format software developers use to write code. The only differences between code and text is the content and file extension. Plain text editors are not exclusive to writing code or reading data; people can write fiction in plain text (and some do).

Reading: Gil, Alex. 2015. "The User, the Learner and the Machines We Make." *Minimal Computing: A Working Group of GO::DH.* May 21, 2015. *This is the canonical essay on minimal computing in digital humanities. Minimal computing centers around the question, "what do we need?" Scholarly communi-*

cations librarians need resources for publishing academic texts online in order to be discovered and accessed by the public without restrictions. This can be accomplished with an open source static site generator, basic web hosting, and minimal maintenance costs. Importantly, the essay asks us to "displace [our] reliance on 'user friendly' mechanisms," like content management systems and hosted platforms, for smaller technical infrastructures, like plain-text and static websites, that are cheaper to sustain and easier to preserve.

Activity:

- Read "Getting Started with Markdown" to learn the basic syntax
- Write a markdown version of your resume/CV in a text editor (such as Dillinger)
- Convert your resume/CV from markdown to HTML5 with Pandoc
- Save the HTML5 code to a file called `resume-cv.html`

Tutorials:

- Tenen, Dennis, and Grant Wythoff. 2014. "Sustainable Authorship in Plain Text Using Pandoc and Markdown." *Programming Historian*, March. *I **strongly** recommend some familiarity with Pandoc for roles or projects involving digital publishing. It was developed with academic writing in mind and is usually a behind-the-scenes piece of software within many digital publishing tools. If you'd prefer not to install Pandoc on your machine, you can use the Try Pandoc online tool to convert between plain-text formats.*

- "Fundamentals: YAML & Markdown." 2020. In *Quire: Multiformat Book Publishing.* J. Paul Getty Trust, Los Angeles. *This is a chapter from the manual for Quire, a multiformat book publishing program, but it is an excellent introduction to the fundamentals of Markdown and YAML in the context of a static site generator for book publishing. The concepts covered in the chapter are essential to using any static site generator.*

Static Websites and Accessibility

Static site generators can be a good tool for learning about web accessibility. They give you full control over the HTML templates and CSS styles. Most of the time, the templates and styles are by a theme a theme. Themes are separate components for static site generators that users make and publishing as open source projects. Whether you are using a theme or creating your templates from scratch, it is your responsibility to ensure that the resulting web publication can be used by everyone.

From an authoring standpoint, plain text is a more accessible format than rich text. Seo and McCurry (2019) study the accessibility of authoring tools for scientific documents (i.e. documents that require math formulas, embedded graphics, figures, and bibiliographic citations) and introduce the Accessible RMarkdown Online Writer, a web-based authoring tool for blind and low-vision writers of scientific content. Markdown, as an accessible authoring format, is a semantically rich plain text format with a minimal syntax, support for LaTeX math, and wide range of output formats: HTML, PDF, Microsoft Word, RTF, EPUB, PowerPoint, etc.

Markdown is independent of static websites and static site generators. Markdown can be use in content management systems and dynamic website platforms, too. Regardless of your web technology stack, distributing multiple formats of your content to your audience, such as Markdown *in addition to* HTML or PDF, provides choice for people to access your work. With static site generators –and web-hosted git repositories– distributing the Markdown source of works is trivial.

Readings:

- "What Is Accessibility?" n.d. MDN Web Docs. Accessed December 3, 2020.
- "HTML: A Good Basis for Accessibility." n.d. MDN Web Docs. Accessed December 3, 2020.
- Seo, Joo Young, and Sean McCurry. 2019. "LaTeX Is NOT Easy: Creating Accessible Scientific Documents with R Markdown." *Journal on Technology and Persons with Disabilities* 7 (16).

Recommending Viewing: Initiative (WAI), W3C Web Accessibility. 2020. "Web Accessibility Perspectives: Explore the Impact and Benefits for Everyone." Web Accessibility Initiative (WAI). December 3, 2020. https://www.w3.org/WAI/perspective-videos/.

Activity

1. Using Chrome Developer Tools, visit a website and perform an accessibility audit.
2. With the same website, use WAVE to perform an accessibility audit.
3. What were the differences between each tool's evaluations?

Static Websites in Context

Learning Objective: Recommend static site generators options to students and faculty in various disciplines.

Popular Static Site Generators

All static site generators have a similar workflow: (1) install the software, (2) add your content, (3) run the `build` or `serve` command from your terminal, (4) upload the files to a server. That said, some static site generators are easier to use than others. In my experience with open source, the popularity of the tool correlates with its easy of use. Here's a very incomplete overview of some popular static site generators.

Jekyll If modern static site generators were a band, Jekyll would be a founding member. It was designed as a blogging system for developers that grew into a more general-purpose website builder over time. For a while, it was the easiest static site generator to use on the GitHub Pages hosting service. This made it an obvious choice for scores of personal blogs, portfolios, documentation websites, and research projects.

Jekyll is written in Ruby. You don't need to know anything about Ruby to use Jekyll, but you do need Ruby and Bundler, a Ruby package manager, installed on your computer in order to use it. This is a big hurdle for installing and using Jekyll on Windows (it can be done). Luckily, there are some resources available to help you make a Jekyll website without installing Jekyll on your computer (a more complete list can be found here).

- Jekyll Now is a tutorial that teaches you how to create a blog "in 30 seconds" through the GitHub web interface. If you ever need a quick and easy place to host your CV or porfolio, Jekyll Now is probably the easiest option available.
- Collection Builder is a Jekyll-based or GitHub-based workflow for creating digital collection and exhibit websites following collections as data principles. The GitHub Pages version can used without installing Jekyll on your machine. Websites with Collection Builder can be made with a spreadsheet of metadata, a folder of images, and a configuration file. Many of the digital collections from the University of Idaho were made with Collection Builder.
- Wax is a Jekyll theme and digital production toolset for making digital exhibits with an interoperable IIIF image viewer. Wax requires Jekyll installed on your computer, but Wax provides everything necessary for transforming a spreadsheet of metadata and folder of images into a digital exhibition website. Visit the documentation to view examples.

Hugo Hugo is a newer static site generator. It was designed for scale and flexibility in order to support any type of website. It is written in the Go programming language, but –like Jekyll– programming knowledge or experience is not necessary unless you need to modify or develop themes or templates. Hugo is known for its speed. Static websites are usually built entirely at once, where as dynamic websites can build pages as needed. When you're working with large static sites, the build time can minutes, which can slow down the development process. Hugo is able to build a website with thousands of pages in seconds – if not milliseconds – while another static site generator might take minutes. I recommend Hugo because it's easy to install and works with a lot of themes.

- Quire is a multiformat book publishing tool by the Getty. It uses Hugo as an underlying static site generator for building the web versions of academic books and museum catalogs. Quire also makes EPUB, PDF, and

Mobi/Kindle files as output formats for the books from Markdown and YAML content. Here's a list of books made with Quire.

- Blogdown is a method of making static websites that contain reproducible data analysis, tables, figures, graphics, and other forms of scientific content generated with the R programming language. This can be an especially useful –and perhaps familiar– option for anyone using R for statistical computing in the sciences and social sciences. Like Quire, Blogdown uses Hugo as an underlying software program for generating the static website files.
- Wowchemy is an open source suite of Hugo website themes and features for educational, personal, and group websites. They provide a variety of templates and guides for new users.

Recommendations for Math

Static websites can render mathematical equations in the web browser using MathJax. MathJax can be added to any website by adding a line or two to the `<head>` element in the HTML template (see: instructions). MathJax is a popular feature included in themes for static site generators focused on academic or technical content. For more specialized cases, these open source projects produce static websites for mathematical content.

- Bookdown, like Blogdown, was developed for users of the R programming language. Unlike Blogdown, Bookdown does not use Hugo for statis site generation, but it does generate static files for website, PDF, EPUB, LaTeX, and Microsoft Word versions of books. This is a popular tool for books on statistics, data science, and R programming.

The following projects do not use Markdown as an input format.

- LaTeXML is a LaTeX to HTML converter. This was used to create the Digital Library of Mathematical Functions of the National Institute of Standards and Technology. This is a great recommendation for scholars who prefer wrtiting in LaTeX or TeX than Markdown. Given that, it relies on a lot more software to process mathematics, TeX environments, macros, and PostScript graphics than a modern static site generator.
- PreTeXt is an XML scheme for research articles, textbooks, and monographs. PreTeXt can create websites, PDFs, EPUB, and Jupyter Notebook documents from XML using XLST. It's currently used to make open mathematics textbooks.

Open Infrastructure for Scholarly Communications

Learning Objective: After examining this topic, participants will be able to identify challenges and take action toward developing and sustaining an open infrastructure for scholarly communications.

The acquisitions of the Social Science Research Network (SSRN) and bepress by Elsevier revealed a "need for community-based scholarly communication infrastructure" by many within the academic library community. SSRN is preprint repository for social sciences and humanities research. Bepress began as an open access, academic journal publishing platform provider and eventually expanded its product list to include a hosted institutional repository platform. Both SSRN and bepress provide scholars with open access alternatives to exlusively publishing their work in subscription-access journals. Hundreds of academic libraries around the world subscribed to bepress products to support their scholarly communications iniatives. For academic library customers, the news of the bepress acquisition in particular meant that their primary vehicle for open access publishing was now owned by one of the largest commercial scientific journal publishers in the world.

This led to the "2.5% Commitment," a famous call to action by David W. Lewis, Dean of the IUPUI University Library at the time. In it, Lewis urges libraries to contribute 2.5% of its total budget to "support the common infrastructure needed to create the open scholarly commons… Collectively we would take responsibility for curating and preserving the world's scientific, scholarly, and cultural heritage thus making it discoverable and freely available to everyone in the world now and in the future" Lewis, 2017. Open source software is a key example of the types of contributions academic libraries can make toward this goal.

Readings:

- "Elsevier Acquisition Highlights the Need for Community-Based Scholarly Communication Infrastructure." 2017. SPARC. September 6, 2017.
- Lewis, David W. 2017. "The 2.5% Commitment." Working Paper. https://doi.org/10.7912/C2JD29.
- Skinner, Katherine. July 23, 2019. "Why Are So Many Scholarly Communication Infrastructure Providers Running a Red Queen's Race?." *Educopia Institute.*

Discussion Questions:

- Are static website technologies examples of open infrastructure for scholarly communication?
- In what ways do static websites address or complicate the impediments to sustaining scholarly communication resources Skinner (2019) outlines?

Community Engagement

There are several vibrant communities working in library, cultural heritage, and scholarly communication technologies who share the values of community-driven, open infrastructure. Here are a few for scholarly communications and digital scholarship librarians to begin following and contributing to this work:

- Invest in Open Infrastructure is “is an initiative dedicated to improving funding and resourcing for open technologies and systems supporting research and scholarship.”
- Code4Lib is a volunteer network of people working in –or adjacent to– library technology.
- Library Publishing Coalition is “an independent, community-led membership association of academic and research libraries and library consortia engaged in scholarly publishing.”
