


Minimizing Computing Maximizes Labor

Quinn Dombrowski <qad_at_stanford_dot_edu>, Stanford University  <https://orcid.org/0000-0001-5802-6623>

Abstract

This paper focuses on the practical realities of implementing the most common minimal computing methods for web development. It addresses the amount of technically-oriented detail-work required to configure the complex set of components that underpin widely-used platforms for static site generation. When going “minimal” requires a great deal of technical labor, what implications does that have for its adoption by scholars without ongoing technical support, or the money or connections to delegate that work? What is the added emotional labor for technical collaborators who are only allowed to “consult” with scholars, when they attempt to coach scholars through implementing a minimal computing site on their own? What opportunities are lost when a web development methodology cannot reasonably be taught in a hands-on way in a short workshop? After describing these challenges, this paper proposes that an infrastructural intervention — encompassing everything from writing better documentation, tutorials, and templates, to developing interfaces that mediate between users and technical complexity — is necessary to realize the potential of minimal computing as a framework for sustainable websites.

Introduction

The conversation within the broad international digital humanities community around “minimal computing” has raised important questions about resource creation and access in contexts where Internet connectivity cannot be taken for granted and about the sustainability of digital humanities projects that scholars create. Minimal computing has served as an umbrella for discussions about hardware (labor conditions and environmental impact of manufacturing modern devices), networking (use of local area networks), and offline availability of resources. Arguably its greatest practical impact on digital humanities as praxis has been through advocacy for static HTML as a medium for digital humanities projects and websites.

1

This paper focuses on the practical realities of implementing the most common minimal computing methods for web development. It addresses the amount of technically-oriented detail-work required to configure the complex set of components that underpin widely-used platforms for static site generation. When going “minimal” requires a great deal of technical labor, what implications does that have for its adoption by scholars without ongoing technical support or the money or connections to delegate that work? What is the added emotional labor for technical collaborators who are only allowed to “consult” with scholars, when they attempt to coach scholars through implementing a minimal computing site on their own? What opportunities are lost when a web development methodology cannot reasonably be taught in a hands-on way in a short workshop? After describing these challenges, this paper proposes that an infrastructural intervention — encompassing everything from writing better documentation, tutorials, and templates, to developing interfaces that mediate between users and technical complexity — is necessary to realize the potential of minimal computing as a framework for sustainable websites.

2

Website as a Long-term Commitment

The most widely-used methods of developing web-based digital humanities projects have brought with them major drawbacks around upkeep and maintenance. The trend of visually elaborate, interactive, Flash-based websites was

3

short-lived but resulted in a set of digital humanities projects rendered inaccessible and unusable as soon as Flash stopped being widely supported.^[1] While individual database-driven content management systems (CMS) have waxed and waned, CMS have been widely used for building web-based digital humanities projects for over 15 years. From a user perspective, they offer many advantages, typically including graphic user interface-based (GUI) website configuration and customization options, a content editing interface that resembles a web form or word processor authoring environment, and the ability to easily extend the website's functionality through plug-ins, which similarly provide user interface-based (UI) configuration.

The ease with which a user can get a CMS installed and running, particularly with the help of “one-click installers” offered by most commercial web-hosting services, belies the long-term challenges the user will likely encounter around maintenance and upkeep. Any website that relies on executable code in order to display content is at risk of that code being subverted in favor of others’ agendas. While individual projects may be targeted specifically for their subject matter (e.g., projects relating to Tibetan language and culture have become targets for Chinese cyberwarfare activities), more commonly, digital humanities projects fall victim to hackers probing sites running a CMS with known security vulnerabilities [Graham 2019] [Logan 2017]. If those responsible for maintaining a site have not been prompt in applying software patches as soon as they are released, the consequences can be quite severe. Many days or weeks of technical work can be required to restore a site to a previous working state (assuming backups were made) and secure it against future incursions [Velloso-Lyons, Dombrowski, Starkey 2021]. The work of recovering a CMS-based site that has been hacked is not trivial: if the scholar^[2] is not comfortable with restoring MySQL backups and looking for malicious code changes and files, the task will likely fall to someone in their professional network, such as IT, library support staff, or a more technically-proficient colleague. Having a project website temporarily taken offline is a source of some embarrassment, but less-technical PIs who delegate the recovery work are largely insulated from the labor and stress involved in the process. This is, fundamentally, the trade-off at the heart of a decision to build a digital humanities project using a CMS: *someone* will be made responsible for the ongoing burden of patching and updating (and, when necessary, recovering) the entirety of the technical stack underpinning the project, in exchange for less-technical members of the project team being able to work in a relatively familiar, comfortable environment.

It is not surprising then that enthusiasm for building websites using minimal computing is likely to come from more-technical members of the digital humanities community, who have personally paid the maintenance costs in exchange for their colleagues’ ease of use [Gibbs 2015]. A project put on the web as static HTML cannot be compromised or hijacked through the numerous, complex vectors of attack to which database-driven CMS sites are vulnerable. Even if the server credentials are compromised and the static HTML files are removed or defaced, restoring the site is a much simpler process. Static sites have other advantages that are likely to be visible to digital humanities practitioners in IT and library roles. For example, it is easy to make a static site available, with its full functionality, in an offline environment.^[3] Likewise, static sites are much easier to preserve in their entirety for long-term archiving.

The questionable longevity of web-based projects is a common concern, particularly for scholars who are accustomed to thinking of their scholarship through the lens of monographs and articles, given the centuries of library- and publisher-based infrastructure built to support long-term access to scholarship published in those genres [Goddard and Seeman 2019]. Established scholars who begin engaging with digital humanities later in their career frequently model a digital humanities project after a book: a sizable undertaking that one can cleanly *finish*, with the result available for others to reference in perpetuity. A turn towards minimal computing as the default platform for developing web-based projects would seem to serve the interests of both less-technical scholars who want to ensure ongoing access to their digital work and technical partners or support staff, whose capacity for new undertakings diminishes with each project that requires ongoing maintenance. In practice, however, minimal computing web development poses a technical barrier high enough to necessitate both technical support and affective labor, even for scholars with enough baseline technical proficiency to be comfortable working independently in a CMS environment [Lincoln et al. 2022].

The Making of “Static”

For a certain generation of scholar-technologists, “static HTML” evokes memories of writing HTML by hand or with the

4

5

6

7

assistance of GUI-based software such as Dreamweaver. Dreamweaver and similar software provided a framework for template files that generated repeated page elements (header, footer, sidebar), along with options for using a “What-You-See-Is-What-You-Get” (WYSIWYG) interface with buttons for styling and formatting text to construct the site and author content. The workflows that underpin recent efforts under the umbrella of minimal computing typically involve static site generator software configured using plain-text files, content authoring using Markdown, layout using template files with a combination of HTML and Liquid, design typically with SASS, with generation of the final HTML done via the command line, then transferred to a hosting environment using Git. This is a vastly different workflow than using the Dreamweaver WYSIWYG interface (which broadly resembled the modern WYSIWYG UI) for formatting text, hand-coding some HTML in the associated Dreamweaver integrated developer environment, and adding inline styles or referencing one or two CSS files for styling.

To use the most common minimal computing workflow, based on the Jekyll static site generation software, the scholar would need to install the Ruby programming language, along with a number of Ruby gems using the command line, then install Jekyll.^[4] Next, they would need to figure out what changes need to be made to the `_config.yml` file (without breaking it altogether with a stray space character), and start authoring content in Markdown — potentially while modifying the front matter variables and writing additional Liquid and HTML to use those variables in the templates. Making any layout or design changes involves figuring out where these changes have to go within numerous possible SASS files. Finally, the scholar would need to generate the output HTML using the command line, and determine which errors and warnings need to be addressed in order for the site to function and which can be safely ignored. This workflow is daunting even for the type of digital humanities practitioner who is not in a developer role as such but frequently handles other kinds of web development tasks and basic coding, along with project and program management (e.g., roles commonly found in library digital scholarship centers). Basic command-line usage, Markdown, CSS (if not SASS), and HTML are likely to be in the skill set of people in these positions, but the documentation for Jekyll and the Liquid templating language is sparse and lacking in detailed examples that someone new to the framework could draw upon while building their proficiency with these tools. Like with a CMS, Jekyll functionality can be augmented through the use of plug-ins — although the number of plug-ins supported by GitHub Pages, a common, free static-site hosting environment, is limited. While workarounds for the plugin restrictions exist, they impede a straightforward GitHub-based code management and site publication workflow. But even when plug-ins address basic functionality, shortcomings in a technical staff’s proficiency with Liquid make it difficult to implement design change requests or minor modifications to the design and layout of the site. Very specific design preferences are common in digital humanities projects managed by a minimally-technical principal investigator and implemented by people in other roles. While there are thousands of free Jekyll themes available, there is very little standardization in how they are implemented, and a solid grasp of Liquid templates is essential for any extensive modification.

8

Minimal Computing in Digital Humanities Workshops

The sheer number of components that go into setting up a Jekyll site, each of them unfamiliar to people with minimal prior technical experience, make it a poor choice for the 2- or 3-day digital humanities workshops for beginners that are commonly run during the summer or at the beginning or end of terms. The 2019 Slavic DH workshop at Princeton, for example, brought together scholars in Slavic languages and literatures from the US and Europe interested in digital humanities, with particular attention to precarious scholars and those without a local digital humanities community. The four-day event included two days of hands-on technical training in building exhibits of visual resources. For Andrew Janco and me, the co-instructors of the technical components, our first choice was to teach Wax, a Jekyll-based system for building exhibits. However, even a preliminary attempt to break down the process of creating a Wax site into discrete lessons that could fit into the schedule for participants made it clear that, realistically, we would be facing irate colleagues if we took this approach. For a minimally-technical audience, we would spend all of the first day walking them through and troubleshooting what might seem to them like arcane, extremely technical steps, before they saw anything that even vaguely resembled their own research materials in a web browser. It was a recipe for them to walk away concluding that digital humanities was too hard for them, that even something as “simple” as putting together a digital exhibit was impossibly complex.

9

If one of the goals of the past decade in digital humanities has been to make these tools and methods available to work based on materials outside the white, Western, patriarchal and usually Anglophone canon, done by people outside of well-funded research institutions, does the procedural complexity of minimal computing risk undermining those gains? What is the value of a technical approach that offers long-term sustainability, if the work that it would sustain is never built due to insurmountable technical obstacles? In response to these concerns, we ran the workshop using Omeka Classic, and participants were able to start assembling their items and exhibits and customizing their site design, using a graphical user interface, on the first day. A number of participants successfully continued their projects independently after the workshop was over.

10

An examination of the agenda for digital humanities workshops that *have* focused on minimal computing suggests that our decision to abandon Wax as a platform for a beginners' workshop may not have been caused by deficits in our pedagogical capabilities or imagination. Short workshops (less than a week) that include Jekyll-based web development use it as an illustrative example. This includes making a case for "reconcil[ing] cultural analytics and research architectures with struggles for justice" [Gil 2018], using demos of existing minimal computing work as a starting point for developing a research agenda [DH 2016], or arguing for minimal computing as a way to improve reproducibility in the humanities [Grguric 2019]. Hands-on workshops for Jekyll are much rarer, particularly workshops targeted towards beginners. Jekyll was listed as one of the options (along with hardware like Raspberry Pi, which takes another perspective on "minimal") for the week-long DH @ Guelph Summer Workshops in 2017. Keystone DH 2017 also included a practical, if not hands-on, workshop on Jekyll for collaboration [Kashyap and Vargas 2017], but the framing suggests the intended audience was people in technical support roles.

11

Human Labor Underwrites Minimal Computing

Bridging the gap between the website that scholars want to be able to build and what they can easily do with Jekyll requires an intervention that combines technical expertise and emotional labor, in a ratio proportional to how much of the work a scholar must do independently. In the simplest case, a technical collaborator can give a scholar a template (such as the CSV file underpinning item generation in Wax) and walk them through how to fill it out correctly. The scholar then returns the template with the content filled in, and the technical collaborator does the rest of the work: correcting any inconsistencies in the data, cleaning any disallowed characters, installing and configuring Jekyll, and importing the data from the template. Configuring the site design to the scholar's satisfaction may require a few iterations, but all together, creating a Jekyll site using this approach will likely take less than a week of the technical collaborator's time.^[5] However, this level of hands-on development work on a scholar's project may be outside the purview of the technical collaborator's job. For example, when the technical collaborator is based in a library or IT unit, a common means of preventing demand from overwhelming the small number of technical staff is to limit their engagement with projects to "consultation," unless their time is explicitly written into a project grant [Cavanaugh and Dombrowski]. Reimagining the Jekyll web development workflow so that it fits into a series of "consultations" involves a shift from primarily technical labor that can be dispatched efficiently to primarily emotional labor that is much more complex and demanding to manage. Paige Morgan describes this kind of activity in a library context:

12

The people I work with are still substantially teaching themselves to get un-stuck, though I provide some specific problem-solving techniques. In these contexts, the emotional labor I perform for scholars as a digital humanities librarian is partly about offering confidence that they can learn that skill, even if it feels strikingly different from their existing abilities. It's also about transparently disclosing some of the potential obstacles they might face, and which are hard for newcomers to imagine in advance. My goal is to underscore that obstacles, like bugs, are par for the course; and also, to give people a sense of what digital work involves so that they can decide whether it's something that they want to pursue. I'm not interested in being an evangelist. [Morgan 2016]

13

In the case of minimal computing consultation, it is difficult to avoid an undercurrent of evangelism. A model where the scholar passes off their project to a technical collaborator can make the use of minimal computing tools a guaranteed outcome, but an approach where the scholar does the work themselves requires ongoing coaching, consoling, and reassurance with each step, each error message, and each roadblock, whether it's dependency conflicts, SASS

14

challenges, variables behaving in surprising ways, or struggles with complex formatting in Markdown. There is a point at which the scholar can work independently on the project, barring any new functionality requests or divergent data, but until the scholar reaches that plateau, there remains a real possibility that the scholar will one day ask a colleague how they built their website, and adopt that tool like WordPress instead, or simply write off their digital project — if not digital humanities more broadly — as a frustrating time sink. Unless the scholar has invested so much time in configuring Jekyll and importing their data that they can't possibly imagine starting over (even along an easier route), there is a risk that the scholar will stop scheduling follow-up meetings and replying to emails, only to re-emerge a year or two later seeking help with fixing their hacked WordPress site. Even when a project is successfully implemented, an ongoing sense of technical unease on the scholar's part can stand in the way of their ongoing engagement and contribution to the project. Despite buy-in from the Scholars' Lab community on a Jekyll-based site relaunch [Visconti 2019], ongoing uneasiness with the interface has reduced the number of website contributions by community members who have run into technical difficulties in the past and who feel like they "should" understand the technology after reading the documentation but continue to struggle with making it work [Visconti 2021].

There are development approaches besides all-or-nothing. In some cases, even under a strict "consultation only" model, a technical collaborator will simply "take care of" part of the work for a scholar, rationalizing that it would take less time and work better for everyone than trying to talk the scholar through doing the steps themselves. But this kind of "shadow labor" has pernicious long-term implications for the project. Under a model where the technical collaborator is responsible for most or all the technical work, they are likewise responsible for the technical maintenance and updates (from which minimal computing workflows are not wholly immune). Having a technical collaborator "take care of" part of the project creates a gap: the scholar walks away without understanding how that part of their project works, only relieved that it *does* work. This kind of hands-on "help" is usually undertaken with the intention of it being a one-time occurrence, but without the knowledge to manage that piece of the project or anyone else to turn to, the scholar will inevitably return seeking more help. Even more emotional labor is required to navigate those relationships and reassert boundaries after having stepped over the line once. While this scenario is hardly unique to projects that use minimal computing methods, the complexity and opacity of the technical components underpinning those technologies make it more likely for this kind of scenario to arise.

15

Articulating the Problem

Minimal computing is offered as a solution to multiple problems, both distinct and overlapping [Sayers 2016]. The environmental costs of server farms and the e-waste produced by continually upgrading computational hardware are one set of issues. Building a website using minimal computing methods may be part of a principled stand against the environmental costs of the digital world, but the ecological differences between a static website and a database-driven CMS are dwarfed by the impact of other personal decisions, which are themselves dwarfed by the impact of decisions made by governments and large corporations. Accounting for the additional time and effort that goes into learning how to build a "minimal" website rather than using a database-driven CMS, one could have a greater environmental impact by simply building a WordPress website and dedicating time to environmental activism instead of debugging Jekyll.

16

If the primary concern is a techno-aesthetic experiment, stripping down common tools and processes to the minimum of what we actually "need" [Gil 2015], then minimal computing is a highly suitable set of tools with which to prototype. Similarly, minimal computing can be one response to a sense of unease with the way that web authoring and publishing interfaces can alienate creators from the technical components underpinning their work. In the context of collaboration, it is important that this sense of unease be shared if it is used to justify the use of minimal computing methods. A technical collaborator who cites concern about a scholar's alienation from their digital tools as the primary justification for imposing a complex minimal computing workflow runs the risk of being written off as an ideologue. While it is true that the individual skills that go into building a Jekyll site are reusable in other contexts, in contrast to the tool-specific knowledge that comes from working within a CMS [Gil 2019], it is not unreasonable to argue that all those skills should not be a prerequisite for publishing one's scholarship online.

17

If the primary concern is doing digital humanities work that can be made available to people who don't have access to high-speed Internet, reliable connectivity, or high-performance hardware, this requires a different lens. Some kinds of

18

graphically intensive virtual reality and augmented reality digital humanities work is out of consideration. Desktop software that is crucially dependent on cloud computing for computationally intensive processing (e.g. Transkribus for handwritten text recognition) should be ruled out. But even if we exclude anything involving specialized peripherals or high-end hardware, along with any software that needs to be installed to run, we are left with so much more than static HTML websites. WordPress, for instance, can be set up in a self-contained way on a USB stick or hard drive, which could be loaded full of digitized texts and images.^[6] Voyant Server, a GUI-based suite for text exploration and analysis, can similarly be run without Internet access or cutting-edge hardware. “Portable” versions of Python and Jupyter Lab make it possible to do more complex and customized forms of computational text analysis without local installation or internet connectivity. Language models can be trained using more robust computational resources and included for use in offline environments. Depending on the number of texts in question, even medium-scale computational text analysis is feasible with a modest laptop.

Creating web-based resources that do not require ongoing upkeep to remain functional — ideally, while minimizing custom code that itself can be a maintenance liability — is a holy grail of digital humanities [Nowviskie and Porter 2010] [Smithies 2019]. The prospect of spending a great deal of time building a website that will likely decay to the point of being unusable in five or ten years is appalling, particularly when compared to the expected longevity of physical books stored in a library. While some updates to the static site generation software itself may be needed if a scholar wishes to continue updating their site, the static HTML websites generated by such software are likely to be viewable as originally intended, even a decade or more later. An interest in preserving digital work in its original form may be the issue that drives scholars towards compromises on “bells and whistles” of website design, indirectly getting to the technical-aesthetic question of “What is the least amount that we need?” through the framing, “What is the most we can do that can still be preserved?” Technical collaborators could successfully advocate for simplifying the design and number of infrastructural components of a website in the name of preservation [Weingart and Lincoln 2019]. The bare-bones authoring and finicky site configuration experiences offered by current minimal computing web tools is harder to justify. The goal of preservation is served by the creation of a static HTML website, and configuration stored in plain-text files rather than a database, but no added virtue is accrued through manual human authoring of those files or content originally authored using Markdown. Using static website generation software like Jekyll is already a compromise that brings with it dependencies, version conflicts, and added configuration hassle, in exchange for being able to do some things more easily than doing the equivalent with hand-written HTML. To enable more scholars to take advantage of the sustainability benefits of minimal computing websites, without passing a large amount of technical and/or emotional labor onto technical collaborators, it is time to prioritize infrastructural interventions that lower the barriers to minimal computing.

19

Maximizing Infrastructure for Minimal Computing

Digital humanities initiatives that have self-identified as “infrastructure” (including Project Bamboo and DARIAH) have a reputation for being large, expensive endeavors that struggle to articulate their goals, let alone accomplishments, in a way that scholars can identify the value they bring [Dombrowski 2014]. A broader conceptualization of infrastructure is needed here, to encompass work that does not exclusively, or even primarily, serve the scholarship-driven goals of a specific project, but supports and enables the work of an arbitrary number of groups.

20

Infrastructure need not be primarily technical. Descriptions and flowcharts about where minimal computing approaches make more or less sense would be extremely valuable to help steer scholars to a suitable technical path. Step-by-step tutorials, particularly ones written without any assumption of prior technical knowledge, can make a tremendous difference in facilitating scholars’ independent use of minimal computing practices. The *Programming Historian* tutorial, “Building a Static Website with Jekyll and GitHub Pages” [Visconti 2016], is one of the few examples of this kind of resource for Jekyll, and it is a vast improvement over general-purpose Jekyll documentation with its detail and clarity.

21

Another highly scalable form of infrastructure is detailed documentation, ideally using discipline-relevant examples. The official Jekyll documentation is notable for its brevity. It largely depends on pointers to the documentation for Liquid, which is heavily shaped by e-commerce examples on account of its original creators: Shopify. Translating examples between domains is a skill that requires practice and experience, and it is demanding to ask scholars to both absorb a

22

new syntax and imagine how it would apply in their project when given only examples based on digital shopping carts. Basic Jekyll and Liquid documentation designed for an intrepid but beginner digital humanities audience, using typical forms of scholarly projects as the basis for examples (as in [Visconti and Walsh 2020]), would be very useful.

Ed (<https://elotroalex.github.io/ed/>) and Wax (<https://minicomp.github.io/wax/>) are themes with additional built-in functionality to make it easier to build an online edition and exhibit, respectively. They represent another kind of infrastructure for doing minimal computing projects, but are both critically lacking in detailed documentation for how to customize them for use with one's own data. It is unclear which configuration options *must* be modified for a new project, which settings *must* be left alone, and in what circumstances one might want to make other modifications to the code. These projects, along with Jekyll plug-ins, multilingual configurations [Lincoln 2021] [Lincoln and Walsh 2021], and other themes and snippets, would benefit tremendously not only from individual documentation, but also from write-ups in the style of Miriam Posner's "How Did They Make That?" blog posts [Posner 2013], breaking down and explaining the configuration used by sites that combine these features in different ways.

23

Finally, there is a need for better technology to reduce the friction that many users experience when building minimal computing websites. There are multiple places where the kinds of small errors that humans are prone to — like putting the wrong number of spaces before a value in a `_config.yml` file, or incorrectly formatting the file name of a blog post to omit the date, or failing to escape one of the many characters that cause problems in Jekyll configuration files or front matter — can lead to frustrating and unexpected systemic failure. A graphical interface for authoring posts and managing site configuration, such as that offered by Netlify CMS,^[7] would ensure greater consistency around those formatting quirks that Jekyll is sensitive to and could offer a more comfortable WYSIWYG environment. A system comparable to WordPress in terms of the affordances it provides users for authoring and configuration, but with the data-handling and templating capabilities of Jekyll, along with Jekyll's static output and sustainability benefits, would be a compelling option for digital humanities web development. Multiple such solutions exist, both as commercial software-as-a-service offerings and a variety of open source projects, but additional work is needed to adapt the code to the particular needs of less-technical digital humanists (e.g., integration with Ed. and Wax-specific configuration settings).

24

Conclusion

The minimal computing movement offers relief to digital humanities developers weighed down by the ongoing hassle of managing and updating WordPress and other CMS sites whose bloated underpinnings, riddled with security holes, offer insufficient value for the cost of their upkeep. Implementing a scholar's project using static site generation software such as Jekyll can vastly simplify both development and maintenance. However, for scholars who do not have access to a technical collaborator who can take care of the multiple distinct technical components, very precisely configured, that are required to create a Jekyll website, the calculus is more complicated. A WordPress site is more likely to offer a familiar interface,^[8] and the ease of a "one-click installer" makes the complexity of Jekyll a much harder sell. Prioritizing minimal computing — whether for environmental, techno-aesthetic, or sustainability reasons — risks leaving behind scholars who lack the technical expertise or ongoing support to manage multiple facets of technical web development.

25

The sustainability and preservation benefits of minimal computing web development are significant, but for these benefits to be realized — particularly by scholars outside of well-funded research institutions with technical developers on staff and by scholars who are not comfortable reading technical prose in English — the digital humanities community needs to engage with building infrastructure to support that work. There are major deficiencies in current documentation, tutorials, and examples for minimal computing web development in digital humanities. Remedying these deficiencies requires clarity in writing, more than deep technical expertise. Scholars with a small amount of hard-won experience with Jekyll could have an impact by documenting their own processes — although this too requires labor without a corresponding immediate payoff for one's own work. Building graphical interfaces that provide a more familiar authoring environment while offering an error-resistant means of configuring Jekyll sites would require non-trivial technical development, even starting from existing open source code. We will know we have built enough scaffolding when minimal computing web development can be taught to technical novices in one- or two-day workshops, with participants walking away comfortable with their ability to continue working independently on their projects. Until then, minimal web development, with all its associated benefits, is a luxury available to those with technical expertise, or the

26

necessary money or connections to absorb the additional labor that this approach requires.

Notes

[1] See [Fiadotau 2020] for a discussion of how one digital gaming community has dealt with long-term sustainability and access issues with Flash.

[2] In this paper, “scholar” is used to refer to an individual whose primary contribution to a digital humanities project relates to their subject-area expertise, in contrast to “technical collaborator” whose contribution involves significant technical work. These terms are meant to relate only the person’s role within the project’s division of labor, recognizing that the “technical collaborator” may themselves have the expertise of a scholar, and may even play the “scholar” role opposite a different “technical collaborator” in a different project context. Projects implemented by a single person who wears the “scholar” and “technical collaborator” hats with equal ease also exist but are not under discussion here.

[3] Contrast, for instance, the work involved in putting a set of static HTML files and associated images onto a USB drive and the site configuration changes, link changes, and archiving efforts that go into doing the same for a Drupal site, as described in [Stevenson 2020].

[4] The Jekyll Quickstart page provides a breezy overview, with a link to a similarly brief Prerequisites page for readers struggling to install Ruby, RubyGems, GCC, and/or Make. There is a more detailed tutorial in the GitHub Pages documentation.

[5] Note that this time estimate is only for the initial creation of the site. Subsequent changes and updates are inevitable, and each requires the technical collaborator to interrupt their current project to make time for this work. A single change for a single project may not have a huge time demand associated with it, but the accumulation of these requests for multiple projects can pose a significant barrier to undertaking new work. Recent approaches to digital humanities project management have worked to constrain this [Sutton Koeser 2019].

[6] Offline, WordPress even offers certain advantages, such as reducing the relevance of security updates. Online hackers cannot exploit vulnerabilities they cannot access.

[7] For an example of a deployment of Netlify CMS in a digital humanities context, see <https://shakespeare-vr.library.cmu.edu/>.

[8] Recent trends in WordPress UI/UX, both in the CMS core with the “Gutenberg” editor as well as more elaborate plug-ins such as the “Divi Builder,” have begun to upend the familiarity of the traditional WordPress interface.

Works Cited

- Cavanaugh and Dombrowski** Cavanaugh, Erica and Quinn Dombrowski. “Personal Entanglements and Graceful Relationship Degradation in Collaborative Projects.” In *Revealing Meaning: Feminist Methods in Digital Scholarship*, edited by Kim Martin and Heather Froehlich. Waterloo: Wilfrid Laurier University Press, forthcoming.
- DH 2016** “Minimal Computing: A Workshop.” *DH 2016 Book of Abstracts*, 2016. <https://dh2016.adho.org/abstracts/304>.
- Dombrowski 2014** Dombrowski, Quinn. “What Ever Happened to Project Bamboo?” *Literary and Linguistic Computing* vol. 29.3 (September 2014). <https://doi.org/10.1093/lc/fqu026>.
- Fiadotau 2020** Fiadotau, Mikhail. “Growing Old on Newgrounds: The Hopes and Quandaries of Flash Game Preservation.” *First Monday* vol. 25.8 (2020). <https://doi.org/10.5210/fm.v25i8.10306>.
- Gibbs 2015** Gibbs, Fred. “Editorial Sustainability and Open Peer Review at Programming Historian.” *DH Commons* 1 (2015). <https://web.archive.org/web/20180713014622/http://dhcommons.org/journal/issue-1/editorial-sustainability-and-open-peer-review-programming-historian>.
- Gil 2015** Gil, Alex. “The User, the Learner and the Machines We Make.” *Minimal Computing: A Working Group of GO::DH*, May 21, 2015. <http://go-dh.github.io/mincomp/thoughts/2015/05/21/user-vs-learner/>.
- Gil 2018** Gil, Alex. “Minimal Computing, Border Technologies and Other Marginal Practices in DH.” *Price Lab Mellon Workshop*, 2018. <https://pricelab.sas.upenn.edu/events/price-lab-mellon-workshop-alex-gil/>.
- Gil 2019** Gil, Alex. “Design for Diversity: The Cases of Ed.” *The Design for Diversity Learning Toolkit*, 2019. <https://des4div.library.northeastern.edu/design-for-diversity-the-case-of-ed-alex-gil/>.
- Goddard and Seeman 2019** Goddard, Lisa and Dean Seeman. “Negotiating Sustainability: Building Digital Humanities Projects that Last.” In *Doing More Digital Humanities*, edited by Constance Crompton, Richard Lane, and Ray Siemens, 38-57. Philadelphia: Routledge, 2019.

- Graham 2019** Graham, Shawn. *Failing Gloriously and Other Essays*. Lincoln, NE: The Digital Press at the University of Nebraska, 2019. <https://thedigitalpress.org/failing-gloriously/>.
- Grguric 2019** Grguric, Eka. "Minimal Computing: One Approach to the Challenge of Computational Reproducibility." *Open Scholarship in Practice*, October 25, 2019. <http://dx.doi.org/10.14288/1.0387127>.
- Kashyap and Vargas 2017** Kashyap, Nabil. and Roberto Vargas. "Collaborative Jekyll." *Keystone DH*, 2017. <https://keystonedh2017.sched.com/event/B3bt/w5-collaborative-jekyll>.
- Lincoln 2021** Lincoln, Matthew. "Technical Tutorial on Translation Links." *The Programming Historian* GitHub Wiki. Accessed September 29, 2021. <https://github.com/programminghistorian/jekyll/wiki/Technical-Tutorial-on-Translation-Links>.
- Lincoln and Walsh 2021** Lincoln, Matthew and Walsh, Brandon. "Technical Tutorial on Setting Up a New Language." *The Programming Historian* GitHub Wiki. Accessed September 29, 2021. <https://github.com/programminghistorian/jekyll/wiki/Technical-Tutorial-on-Setting-Up-a-New-Language>.
- Lincoln et al. 2022** Lincoln, Matthew, Jennifer Isasi, Sarah Melton. "Relocating Complexity: *The Programming Historian* and Multilingual Static Site Generation." *Digital Humanities Quarterly*, forthcoming.
- Logan 2017** Logan, Jim. "A Pioneering Netizen." *UC Santa Barbara News*, January 10, 2017. <https://www.news.ucsb.edu/2017/017526/pioneering-netizen>.
- Morgan 2016** Morgan, Paige. "Not Your DH Teddy-Bear; or, Emotional Labor is Not Going Away." *DH + Lib*, July 29, 2016. <https://acrl.ala.org/dh/2016/07/29/not-your-dh-teddy-bear/>.
- Nowvieskie and Porter 2010** Nowvieskie, Bethany and Dot Porter. "The Graceful Degradation Survey: Managing Digital Humanities Projects Through Times of Transition and Decline." *DH 2010 Book of Abstracts*, 2010. <http://dh2010.cch.kcl.ac.uk/academic-programme/abstracts/papers/html/ab-722.html>.
- Posner 2013** Posner, Miriam. "How Did They Make That?" *Miriam Posner's Blog*, August 29, 2013. <https://miriamposner.com/blog/how-did-they-make-that/>.
- Sayers 2016** Sayers, Jentery. "Minimal Definitions." *Minimal Computing: A Working Group of GO::DH*, October 2, 2016. <http://go-dh.github.io/mincomp/thoughts/2016/10/02/minimal-definitions/>.
- Smithies 2019** Smithies, James et al. "Managing 100 Digital Humanities Projects: Digital Scholarship & Archiving in King's Digital Lab." *Digital Humanities Quarterly* vol. 13.1 (2019). <http://www.digitalhumanities.org/dhq/vol/13/1/000411/000411.html>.
- Stevenson 2020** Stevenson, Karen. "Sending a Drupal Site Into Retirement." *Lullabot*, February 13, 2020. <https://www.lullabot.com/articles/sending-a-drupal-site-into-retirement>.
- Sutton Koeser 2019** Sutton Koeser, Rebecca. "Document ALL the things!" *The Center for Digital Humanities at Princeton*, August 12, 2019. <https://cdh.princeton.edu/updates/2019/08/12/document-all-things/>.
- Velloso-Lyons, Dombrowski, Starkey 2021** Velloso-Lyons, Mae, Quinn Dombrowski, and Kathryn Starkey. "The Global Medieval Sourcebook: Creating a Sustainable Digital Anthology of Medieval Texts and Translations." *Seminar: A Journal of Germanic Studies* vol. 57.3 (2021). <https://doi.org/10.3138/seminar.57.3.1>.
- Visconti 2016** Visconti, Amanda. "Building a Static Website with Jekyll and GitHub Pages." *The Programming Historian*, 2016. <https://programminghistorian.org/en/lessons/building-static-sites-with-jekyll-github-pages>.
- Visconti 2019** Visconti, Amanda. "Website Relaunch!" *Scholars' Lab*, January 15, 2019. <https://scholarslab.lib.virginia.edu/blog/site-relaunch/>.
- Visconti 2021** Visconti, Amanda. Personal communication regarding Jekyll usability. September 13, 2021.
- Visconti and Walsh 2020** Visconti, Amanda and Brandon Walsh. "Running a Collaborative Research Website and Blog with Jekyll and GitHub." *The Programming Historian*, 2020. <https://programminghistorian.org/en/lessons/collaborative-blog-with-jekyll-github>.
- Weingart and Lincoln 2019** Weingart, Scott. and Matthew Lincoln. "CMU Library Labs (2020-2024)," 2019. <https://doi.org/10.1184/R1/13522718>.



