

HOW DID THEY MAKE THAT?



 >> [Journal](#) >> [Issue 1](#) >> [Editorial Sustainability and Open Peer Review at Programming Historian](#)

Editorial Sustainability and Open Peer Review at Programming Historian


How we made it

by fred gibbs



Q and A

Project information



Humanities publications are often maintained by collaborators spanning vast geographic distances and uneven temporalities. Such logistical challenges are not trivial, as anyone who has grimaced at yet another version of a jointly-edited Word document arriving in their inbox can attest. These issues only multiply when creating or maintaining an online journal—especially one with a rolling submission process—constantly managing the collective work of authors, contributors, reviewers, and editors.

[Programming Historian](#) is an online, open access, peer reviewed suite of over 40 tutorials (and growing!) that help humanists (though slanted towards historians) gain experience and insight into not only programming, but a wide range of digital tools, methods, approaches, and workflows to facilitate their research. The success of the site is driven entirely by the energy and enthusiasm of volunteers (general editors, lesson authors, lesson reviewers, etc). Over the last year, we have adopted and refined an open and sustainable editorial and review process centered on [Markdown](#) and [GitHub](#) that may serve as a useful model for online journals.

WHY WE NEEDED A CHANGE

As with many—if not all—volunteer projects, time is always the most limiting factor. Any workflow or publishing platform—no matter how popular or potentially useful—becomes detrimental when small tasks seem like they are taking longer than they should. It was precisely this issue that drove Programming Historian from WordPress to GitHub (and GitHub Pages).

WordPress was a nearly ideal solution for several years. However, as our publishing rate increased and our number of lessons grew, managing lessons (including finished lessons, lessons awaiting editing or reviewing attention, started but abandoned lessons, etc) became too unwieldy. Not only was it difficult to manage versions of lessons and the status of lessons at various stages of production, it was even harder within the framework of the WordPress administrative interface to manage the editorial process as we preferred. Surely we could have written better code when we made customizations over time, but such is the gap between ideal practice and practical reality within entirely volunteer projects.

WHAT WE USE

We first [converted our existing lessons](#) from HTML to [Markdown](#) (using the fantastically useful [Pandoc](#) conversion tool); we now require that our authors submit lessons in Markdown so that they can seamlessly enter our [editorial workflow](#). Markdown is a plaintext syntax that allows for simple text formatting (headings, underline, bold, italic, lists, etc) without all the bulky tags and symbols that make HTML difficult to work with during the editorial process. The plaintext file format means that we don't have to worry about shifting proprietary file formats, or bizarre formatting issues that inevitably arise when different people try to serially edit the same file in different word processing applications.

We now use [GitHub](#) to organize, manage, and version lessons, as well as other site pages and ancillary resources (images, data files, etc). GitHub is a platform used primarily by open source software developers that allows multiple people contributing to a particular project—working from a common code base—to develop sections of code independently without stepping on each others toes (namely, editing outdated versions of files). But GitHub's utility goes far beyond keeping track of code. Its generic functionality to manage versions of files and their revision histories makes it ideal for managing content of an online journal, as well as facilitating open peer review of that content. While it can seem intimidating at first (especially compared to WordPress), excellent documentation makes the barrier to entry quite low.

WHY WE USE IT

Moving to GitHub alleviated the burden of editing lessons (or other site pages) within an online administrative interface and eliminated the need to create separate HTML versions of the lessons. GitHub allows us to work with individual lesson files, offline if we choose, and easily (re)incorporate them into the Programming Historian website with minimal effort. Going from an easily readable file composed in Markdown to an HTML version of the lesson is effortless because all GitHub repositories (where the Markdown files are stored), come with built-in functionality called [GitHub Pages](#) that automatically takes pages written in Markdown (or HTML) and converts them behind the scenes (via [Jekyll](#)) into static HTML pages that comprise the working website. The built-in [templating system](#) is simple, yet highly functional (and designed for non-coders), minimizing our frustration getting our pages to look the way we want. Thus, the easily readable and editable format of our Markdown lessons essentially comprise the website content itself. This way, we have one canonical working version of every lesson, a feature that is essential for us not only during the editing process, but especially as editors and readers try to keep lessons up to date while versions of tools and technologies continue to evolve. There is always only one file in one place to update, and everyone has

access to it via the open [Programming Historian repository](#). If someone needs to see what a lesson looked like at a particular time (per traditional publication expectations), it is readily viewable via GitHub's web interface.

The GitHub platform excels at providing the key functionality we need (an open, versioned repository), which makes it easier for us to focus on our lessons rather than spending time on the website infrastructure itself. GitHub's many tutorials ([like this one](#)) quickly brings new authors and reviewers up to speed—and they can do everything they need to do via the GitHub website. Markdown is very simple to read and edit, making editorial work much easier than dealing with other file formats, whether HTML, Word, or PDF. The file system-like repository structure makes organization a snap. Versions of lessons in various states, which previously had been separate entities to manage, are now simply snapshots viewed in the GitHub history browser. Using GitHub's built-in “issue tracker” (an “issue” is GitHub-speak for something that eventually needs to get done or resolved), any discussions about potential changes to the site are automatically archived for future consultation, which has been utterly invaluable for the editorial team.

COMMUNITY OPENNESS

Employing GitHub as a staging area for lessons-in-progress facilitates our commitment to an [open peer review process](#). Authors who write new lessons contribute them to our repository (via [pull requests](#)) so that lessons are immediately online and accessible for editors and reviewers, but virtually invisible unless one knows about them. Editors then create an “issue” via the issue tracker; peer reviewers (once they've created a free GitHub account) attach their comments to it. As an open, public repository, all reviews are fully visible to the author and everyone else—and the reviewers get proper recognition and credit for the difficult review work. We believe that this transparency results in more open and honest communication, even amidst critical reviews. Our process thus embodies our philosophy that editors, authors, and reviewers can work together convivially to create significantly more useful resources than via secret editorial brokering and reviewer evaluation.

The community response to our move to GitHub was both productive and inspiring. We immediately received (and continue to receive) a continuous stream of suggestions for improvement (via issue tracking or pull requests) from folks who had no stake or involvement in a particular lesson, but have an idea for making it better. All changes are approved by an editor, but such suggested changes can be incorporated into the live site with the click of a virtual button (hence the power of using GitHub to manage live content). Productive discussions about technologies and techniques pertaining to the lessons have unfolded in the

built-in commenting feature of GitHub, a fringe benefit that underscores the value of using an open, community platform for sharing knowledge, whether via formal lessons or informal comments and suggestions.

SUSTAINABILITY

From a sustainability perspective, required resources are minimal: no hosting service to pay for, no platform installations to maintain, no special software required. We remain at the mercy of GitHub's stability, but so much is true for any hosting provider. The site remains entirely exportable and easily portable with a small footprint and few dependencies.

SKILLS AND RESOURCES

- Technical Skills: Basic concepts of version control, familiarity with GitHub's functionality, editing plaintext files with a text editor, Markdown, HTML
- Competencies needed: Willingness to learn and adapt new workflows and processes, technical writing to produce documentation, familiarity with the terminal/command line is very helpful, but not essential
- Infrastructure needed: Free GitHub account
- Resources: [GitHub](#), [Markdown](#), [Introduction to GitHub](#), [Mastering Markdown](#), [Forking a Repository](#), [Using Pull Requests](#), [PH Reviewer Guidelines](#), [Reviewing Lessons with GitHub](#)

AUTHOR

fred gibbs

FIND OUT MORE ➞



DHCommons Journal

[About the journal](#)

[Review guidelines](#)

[Submission guidelines](#)

Site designed by **Agile Humanities** in association with [Intelligent Machines](#).