Website Documentation | Mobile Bay DSA

Last Edited 10/16/2025

Contents

Part I

Hosting

Introduction

Our website is built with Jekyll, Tailwind, and Markwhen; hosted with Github Pages from https://github.com/alabenian/mobilebaydsa; and presented to https://www.mobilebaydsa.org/ for the public to see. This little book exists to help members work on the site and consider any alternatives to the current setup if needed.

8

Chapter 1

Basic Hosting Information

Our domain name `mobilebaydsa.org` is registered through Godaddy and currently presents the Github Pages site at https://alabenian.github.io/mobileba ydsa. It could present any other source, but I chose to use Pages because our Godaddy plan did not come with a site builder and other options would cost some money while Pages was free, and because it gives us complete control over the site as long as it's static (i.e. as long as we don't need to store data on the fly, and all processing can be left to the user's computer, unless an additional outside service is used) and its contents are safe to keep public. Godaddy is told where to get the website using four `A`-type "@" DNS records which give it the IP range of Github and one `CNAME`-type "www" record which tells it to use `alabenian.github.io`:

	Type ⑦	Name ⑦	Data ⑦	TTL ⑦	Delete	Edit
☐	A	@	185.199.108.153	600 seconds	🗑	✎
☐	A	@	185.199.109.153	600 seconds	🗑	✎
☐	A	@	185.199.110.153	600 seconds	🗑	✎
☐	A	@	185.199.111.153	600 seconds	🗑	✎
☐	CNAME	www	alabenian.github.io.	1 Hour	🗑	✎

All this is available from Domain/DNS/DNS Records at the `mobilebaydsa.org` control page in Godaddy; a more in-depth tutorial is available at this Medium page. Part of the original process involved also deleting the DNS records for the fallback site generated by Godaddy.

Github makes things easy to set up on its end: we publish the site to Github as a repository (https://github.com/alabenian/mobilebaydsa), enable Github Pages, which will automatically deploy it to `[username].github.io/[repository name]`, then tell it to use the domain name `mobilebaydsa.org` and enforce HTTPS:

GitHub Pages

GitHub Pages is designed to host your personal, organization, or project pages from a GitHub repository.

Your site is live at https://mobilebaydsa.org/
Last deployed by 🐱 alabenian 2 days ago

☐ Visit site   Unpublish site

Build and deployment

Source

Deploy from a branch ▾

Branch
Your GitHub Pages site is currently being built from the /docs folder in the main branch. Learn more about configuring the publishing source for your site.

⑂ main ▾   📁 /docs ▾   Save

Learn how to add a Jekyll theme to your site.

Your site was last deployed to the github-pages environment by the pages build and deployment workflow.
Learn more about deploying to GitHub Pages using custom workflows

Custom domain

Custom domains allow you to serve your site from a domain other than alabenian.github.io. Learn more about configuring custom domains.

mobilebaydsa.org   Save   Remove
● DNS Check in Progress

☑ **Enforce HTTPS**
HTTPS provides a layer of encryption that prevents others from snooping on or tampering with traffic to your site.
When HTTPS is enforced, your site will only be served over HTTPS. Learn more about securing your GitHub Pages site with HTTPS.

This should then automatically add a `CNAME` document containing "`mobilebaydsa.org`" to the repository. The only other setting I've changed here is to have it deploy from the `/docs/` folder of the repository rather than the root. Here `/docs/` is just a renamed version of the `/_site/` output of Jekyll; this is deployed as a static site pre-built by Jekyll locally rather than one where Github runs Jekyll itself after changes are made to the repository, due to an incompatibility with one of the plugins I was using for our events posts (jekyll-leaflet). *I will explain this more later, but for now, it's not relevant.*

1.1   Github Pages and Potential Alternatives

Github Pages is a nice free way to host the website with maximal control over its contents as mentioned before, but it can be somewhat inaccessible to anyone who isn't already used to working with websites this way. The biggest con besides

this is the fact that, on the free plan, all contents of the site have to be public if an external service is not used to store private, member-only information. Despite the extra difficulty of getting set up for those who would already have access, it's arguably more democratic – changes are published to the site when someone makes changes on their own download of the repository and sends them to Github through a **commit** with details of the changes they made to a local **branch** via the Github Desktop app (or via the command line), before making a **push request** that, when accepted, will publish them to the **main** branch and thereby the official website. The push request can be accepted by the owner (currently "alabenian," but could be transferred to an official Mobile Bay DSA account) or anyone who has been listed as a collaborator on the project. This setup has the benefit of allowing *anyone* to suggest changes to the website to be confirmed by authorized accounts. This way everyone has a voice to publish blog posts, new pages, design ideas, etc. to the site in a way that official collaborators can see every change that has been made and prevent security problems. The site uses a few tools, primarily Jekyll, but is otherwise made from scratch.

It might be better to use a site builder like Wordpress.com which allows easier visual editing and supports blogs, etc. the same as Jekyll. Wordpress is especially nice because it would allow us to include member-only content on the website, secured either by password or a whitelist of Wordpress accounts. Wordpress does not allow free sites to use custom domains, however, and the cheapest plan that *does* is priced at \$96/year. Godaddy has a forwarding feature that would let us send anyone at `mobilebaydsa.org` to the Wordpress site, though I don't know how this would affect the site's presence in Google searches. The other major differences to consider are restrictions on design (which can also be removed with a \$24/year custom CSS add-on) and a Wordpress watermark that comes with the free plan (I've seen this on other DSA chapter sites, and it's not *too* bad).

Part II

Basic Structure & Working on the Repository

Chapter 2

Setup

There are three command-line tools in total used to build the site: **Jekyll**, **Tailwind**, and **Markwhen**. Installing Jekyll takes the greatest priority, as it is necessary to build the site at all (if we switched to having Github build the site with Jekyll itself, installing it locally wouldn't be 100% necessary, but it's still nice to be able to view what you've done before publishing). Tailwind is second, because it creates the `.css` file used by the rest of the site for styling. Because this file has already been created, Tailwind is only necessary for *editing* it. Finally, Markwhen is only used to render and update the timeline used on the Calendar page of the site, and is only necessary to someone who is adding or removing events on the calendar.

Jekyll (https://jekyllrb.com/) simplifies work on the site by structuring the files by root directory markdown (or HTML) files containing the contents of each page, `_layout` HTML files which are referenced in each to determine how the contents are presented, and `_include` HTML files which are referenced in either the layout or content files to display some extra piece of the website (e.g. a navbar or a commonly-occurring button or icon). The resulting file structure of the site will be discussed further in "Repository Structure." It requires Ruby (installation information here) and RubyGems (which should be included with Ruby) to install via `gem install bundler jekyll` in the terminal. After installation, the site can be built by opening the repository folder in the terminal (or `cd`-ing into it) and running `bundle exec jekyll serve`. I've used some additional plugins on the site, so it will throw an error on the first build and prompt you to install those; from there any work with Jekyll should be smooth sailing!

Tailwind (https://tailwindcss.com/) simplifies writing CSS for the site by supplying a set of classes with short and easy-to-remember names. It can be installed as a CLI tool with `npm install tailwindcss @tailwindcss/cli`. (This uses `npm`, so unfortunately you'll need to install Node

first: https://docs.npmjs.com/downloading-and-installing-node-js-and-npm)
To use it on the site, just open the folder in another terminal tab (or, again,
`cd` into it) and run `npx @tailwindcss/cli -i ./styles/initstyles.css`
`-o ./styles/finalstyles.css --watch`. It will then replace the "`@import`
`"tailwindcss";`" in the initial styles file and watch for future changes to
re-run the command in that terminal instance. This adds so much extra hassle
for getting set up to edit styles, I may remove it in the future – it made the
process of designing the site much easier for me because I had already installed
it and knew its classes.

Markwhen renders the markwhen file `calendar_source/main_calendar.mw`
into `calendar_source/timeline.html`, which is embedded on the Calendar
page. Install it using `npm i -g @markwhen/mw` (this also requires Node – see
instructions for Tailwind). After installation, all you need to do is run `mw`
`calendar_source/main_calendar.mw calendar_source/timeline.html` to
update the timeline if changes to `main_calendar.mw` have been made. See the
"Events & Posts" chapter for more information about editing the timeline.