StaticSiteBuilder: A Static Website Builder

Marc Rochkind
mrochkind@gmail.com
21-Feb-2024

Table of Contents

1. The Problem StaticSiteBuilder (SSB) Solves

For about 25 years I've hosted my personal website with a hosting service that ran Apache, with most of its pages coded in PHP. This was the case even when the page was static. For example, I often used PHP as a sort of macro language, coding something:

```
put_image('https://mrochkind.smugmug.com/photos/i-WqB4zkk/0/Th/i-WqB4zkk-Th.jpg',
 'left');
```

instead of:

```
<img src='https://mrochkind.smugmug.com/photos/i-WqB4zkk/0/Th/i-WqB4zkk-Th.jpg'
style='float: left'>
```

I also used PHP to include common elements, such as page footers and site-wide menus. I almost never used PHP for actual server-side processing, such as form processing.

I was paying a little less than $10 a month for hosting, and it occurred to me that I really had a static site that could just as well be hosted on Amazon Web Services Simple Storage Service (AWS S3), at a cost of less that $10 per year. If I could get rid of all that PHP, the site would be much simpler to maintain, too. Over the years my site sometimes broke because my hosting company updated its PHP, and I had to go through and change the code. All that work would go away if the PHP went away.

But one big problem stood in the way: With no server, there would be no server-side includes, and that would mean that every page would have to have its own copy of common headers, footers, and site-wide menus, which is too painful to even consider if they have to be inserted manually whenever they change.

The solution is to process the includes on client side, where I create and edit the site, and to upload self-contained, complete HTML files. The simplest way would to encode includes using existing SSI syntax, like this:

```
<!--#include file="included.html" -->
```

and then write a simple client-side program to process the includes, outputting complete HTML files. I'm certainly not the first person to think of this; for example, see github.com/yktoo/clssigen.

But rather than do just handling includes, I decided to write a much more elaborate program called StaticSiteBuilder (SSB) that would know about the whole site, with these principal features:

- You can create a website with multiple pages connected with a menu without any HTML, CSS, or JavaScript. But if you want to use HTML, CSS, or JavaScript, you can do so without restriction.
- Markdown is built-in.
- Masonry (a grid layout system) is built-in (masonry.desandro.com).
- Sites are responsive, so the they work on large screens or small ones (e.g., phones).
- Created pages automatically are added to the site-wide menu, in a sidebar or, on small screens, as a hamburger menu.
- Page headers, footers, menus, and CSS are automatically included, and you can control their contents if you want.
- Pages can be edited inside the application, or with an external editor.

A few non-features:

- No support for uploading the pages to the host. This is done with a separate application, such as WinSCP, which I use. Or you can use the host's web features (e.g., with an upload button).
- No support for menus other than in a left sidebar on large screens and a hamburger menu on small ones.
- No site preview. However, since no server is needed, you can just open a page on your local file system in your browser. When the page changes, you just reload the browser view.
- No HTML or any other kind of syntax checking, and no support for syntax-aware HTML editing. You just type whatever you want into the file.

(SSB follows the minimalist philosophy of original UNIX, which I worked on in the early 1970s at Bell Labs. The basic idea is that it has just the essential features and if you want it to do more you change the code.)

The SSB source code is on GitHub:

```
github.com/MarcRochkind/StaticSiteBuilder
```

It has an MIT license, which means that you can do anything you want with it. It's about 900 lines of Python in a single file.

I often build simple websites to hold resources for classes I teach, and I've never found a faster way to build them than with SSB. If you want to see my own SSB-built website (not the class website), go to mrochkind.com. You'll see that I use Masonry a lot.

There are other static website builders, such as Jekyll and Hugo, but I haven't used them. They seem to be at a lower level than SSB and are much more complicated to use. I haven't investigated them enough to determine whether they solve the problem that SSB solves.

2. Hosting a Static Website

The files produced by SSB have to be uploaded to the cloud so people can access the site.

2.1. AWS S3

All you need to host a static website is a cloud-based file system that allows files to be accessed with a URL. AWS S3 works this way. You have to make all the pages readable by the public (not the default), and check the option to make the bucket (where the pages are located) a static website. That option allows you to reference the home page by the bucket URL, such as:

```
http://s3.amazonaws.com/MJR-Forrest-2008-01-28
```

instead of having to specify the HTML file, like this:

```
http://s3.amazonaws.com/MJR-Forrest-2008-01-28/index.html
```

S3 costs are so low than my website can be hosted for less than $2 per year (not month!). Unfortunately, only http is supported, not https. To get https, you have also set up AWS CloudFront, which cost about 50 cents a month more. CloudFront also speeds up access, which is its primary purpose. This isn't important for my site, but it might be for yours.

Because of its caching, when you upload a changed page it won't appear right away in browsers. To speed this up you can invalidate the page from the CloudFront console.

If you want your own URL (e.g., mrochkind.com), you set up that with AWS Route 53.

Setting up an S3 site with CloudFront is very complicated, but AWS provides detailed step-by-step instructions. The first time I did it I skipped a couple of steps that I didn't think mattered, and the site didn't work. Then I followed all of the steps exactly, even if I didn't understand what they were doing, and it worked perfectly.

2.2. W3Schools Spaces and GitHub Pages

Two simple, free places to host your SSB are are W3Schools Spaces (w3schools.com/spaces) and GitHub Pages (pages.github.com). All you have to do is upload your SSB pages using their website upload facilities. Last I checked, free W3Schools Spaces sites were limited to 20MB, but GitHub Pages allows up to 1GB. There are surely other places where you can host small websites for free.

If you use GitHub Pages, be aware that the public has access to the raw pages themselves, not just the site. So don't put anything private in any of your HTML or other files.

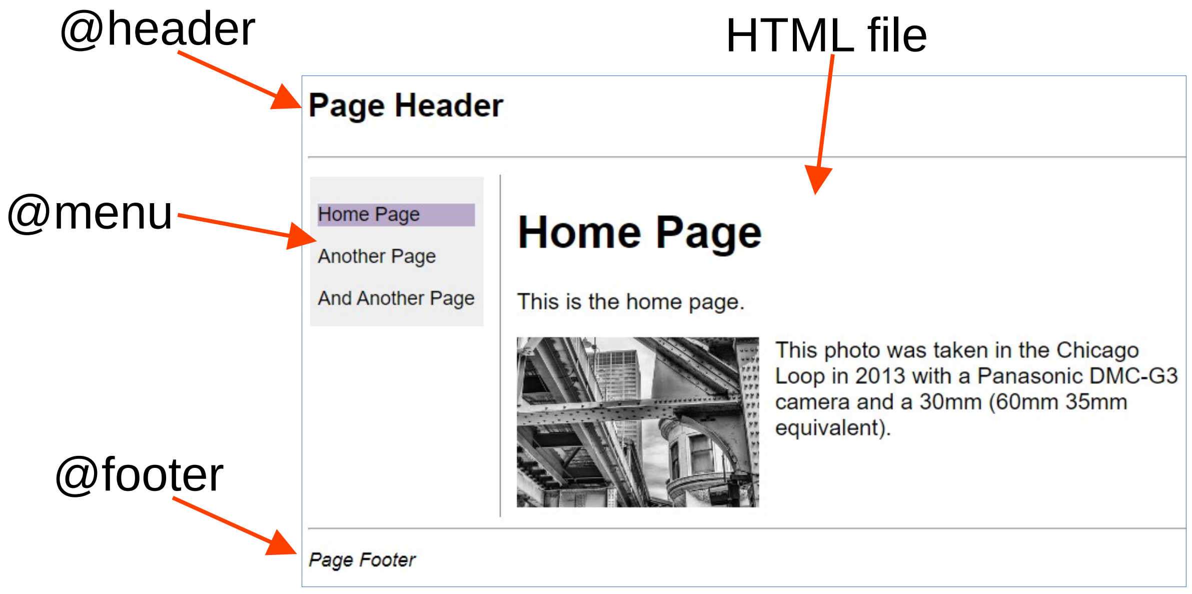
2.3. Other Cloud Drives

Other choices that I haven't tried are Microsoft's OneDrive, Google Drive, and Dropbox. None of them support websites directly. From what I've read, you can use DriveToWeb (drv.tw) for the first two and DropPages (droppages.com) for the third. (It seems that DropPages has a monthly fee that's more than I was paying for full-blown hosting on an Apache server.)

3. Website Structure

Like most sites, an SSB site consists of a collection of HTML files, each of which is a page. Pages are linked to by a menu that appears in a left sidebar on larger screens, and as a hamburger menu on small screens, such as on phones. As explained earlier, the entire page is in the HTML file, including common elements such as the page header and footer and the sidebar/hamburger menu.

An individual page has elements from its own HTML file plus the contents of three other files: @header, @footer, and @menu, as shown in the picture.



Page Header

Home Page

This is the home page.

This photo was taken in the Chicago Loop in 2013 with a Panasonic DMC-G3 camera and a 30mm (60mm 35mm equivalent).

Page Footer

Figure 1: Page structure

There's also a @site.css file that contains CSS that applies to every page. Initially it's empty:

```
#page-footer {
}
#page-header {
}
#menu {
}
#title {
}
#header-hr {
}
#footer-hr {
}
#main {
}
#main p {
}
```

You can populate CSS file as you wish. If you want CSS to only apply to a single page, add a <style> element.

When you create a new page in SSB. it's automatically added to the @menu file. When the site is rebuilt, every page's menu will show the new page. You also have to rebuild the whole site when you change the @header or @footer files.

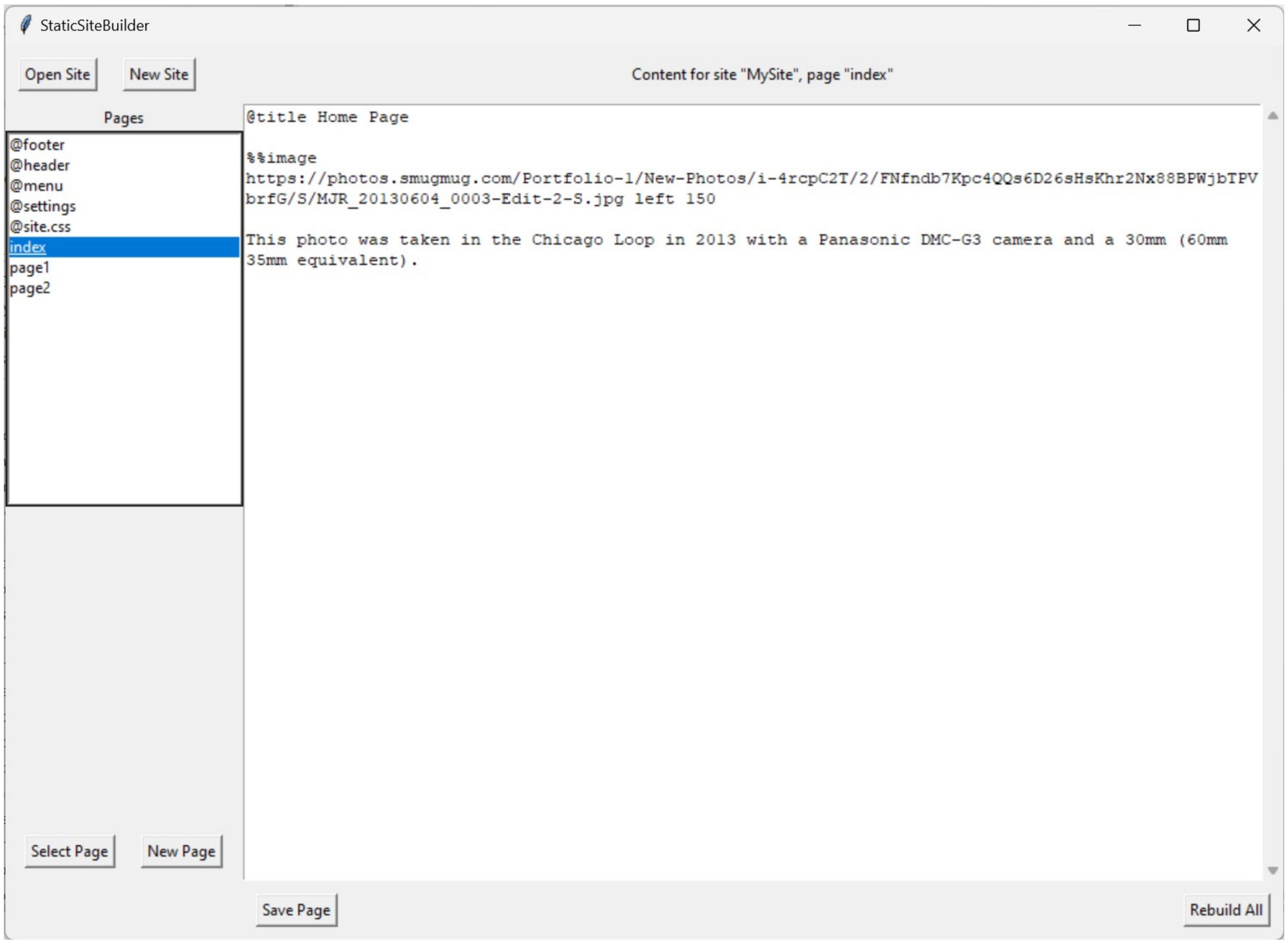
A page's HTML file has an id which you assign when you create the file in SSB. The id is used only internally and never appears on the built site. What the site shows is the page title, which you can change whenever you want. (You then have to rebuild the site, since the title shows in the menu.)

So, as you've gathered, the site has to be rebuilt whenever anything changes that affects more than the page you're editing. Fortunately, rebuilding is very fast. When you upload the site to wherever it's hosted, you upload all of the pages. If you know that no common elements changed, you can get away with uploading just the changed pages. That part is entirely up to you—there's no automatic uploading in the SSB application itself. (All of this rebuilding could be avoided if the server offered server-side includes, but, of course, there is no server.)

The @menu file is normally just a list of ids, and an id is placed there only when the page is first created. You can change the order of ids, remove an id, or add additional links or other HTML to the @menu file as you wish.

4. Using the SSB Application

Figure 2 shows the SSB window.



```
@title Home Page

%%image
https://photos.smugmug.com/Portfolio-1/New-Photos/i-4rcpC2T/2/FNfndb7Kpc4QQs6D26sHsKhr2Nx88BPWjbTPV
brfG/S/MJR_20130604_0003-Edit-2-S.jpg left 150

This photo was taken in the Chicago Loop in 2013 with a Panasonic DMC-G3 camera and a 30mm (60mm
35mm equivalent).
```

Figure 2: SSB window

To edit a page, you select it by its tag and click the Select Page button. Then you can edit it as needed. It's saved when you click Save Page or Rebuild All or select a different page. To rebuild the site, which you do often when first building the site, you click Rebuild All. The button New Page give you a new page; you're prompted to enter its id. You create a new site with New Site. That gets you to an open folder dialog; you have to create a folder for the new site before you can open it. You can open an existing site with Open Site, from which you select that site's folder.

The @settings page contains site-wide settings. There is only one: @prevnext causes each page to have Prev and Next links at the top to allow navigation from page-to-page without having to click on the menu links themselves.

Generated HTML files go directly into the site's folder. The text files you edit are in a sub-folder named "data". You don't usually deal with those directly, as SSB accesses them as needed when you click the Select Page button.

If you prefer, you can edit the text files with an external editor, such as Sublime Text and then click SSB's Rebuild All button to rebuild the site. There's no requirement to edit the pages inside SSB if you prefer not to. You do deal with the data folder if you're using an external editor. Make sure you edit txt files, not html files.

5. Installing SSB on Your Computer

I’ve run SSB only on Windows, but it probably also runs on MacOS or Linux. You might have to make a few adjustments to the code, or maybe not. Here’s how you install it:

- Download the SSB file build.py from GitHub:

  ```
  github.com/MarcRochkind/StaticSiteBuilder
  ```

- In the same folder as build.py, put two JavaScript files that you’ll need if you’re using Masonry:

  ```
  imagesloaded.pkgd.min.js
  masonry.pkgd.min.js
  ```

  You get these from:

  ```
  github.com/desandro
  ```

- Install the markdown module if you don’t already have it:

  ```
  pip install markdown
  ```

- Then run build.py and you should see the SSB window.

6. Coding Pages

You can type anything you want into a page that makes sense for it being treated as HTML, CSS, and JavaScript by a browser. That’s entirely up to you, and SSB provide no help whatsoever, other than some shortcuts, of two kinds: page directives at the top of the page that affect the whole page, and local directives that have a local affect.

These are the page directives, all but the first of which are optional:

@title *title*	Page title as it should appear in the menu and at the top of the page. Required.
@nomenu	No sidebar menu on this page.
@masonry *options*	Layout the page using masonry (masonry.desandro.com). The options are passed when the Masonry constructor is called. For example: `@masonry gutter: 5, columnWidth: 100`
@colors	With masonary, vary the background colors of the cells.

These are the local directives:

%%image *src* [*class* [*maxwidth*]]	Shortcut for: `<img src="src" class="class" style="max-width: maxwidthpx;">` *class* can be anything, but “left” and “right” have their CSS built-in with “float: left;” and “float: right;”, respectively.
%%clear	Shortcut for: `<br clear=all>`
%%cell [*class* [*url*]]	Begins a Masonry cell with class *class*. If *url* is specified, the entire cell is an anchor that goes to that URL.

You can code the page with Markdown. For Markdown instructions, go to these sites:

```
python-markdown.github.io/#Features
daringfireball.net/projects/markdown/syntax
```

If a line has an https URL on it and nothing else, it’s turned into a link. As an example, here’s the start of a page I used for one of my courses:

```
@title Nuclear Power: Best Solution or Worst?

Slides used in class (PDF)
https://mrochkind.com/OLLI/1-Nuclear-2024.pdf

## Class Videos

Moments in NRC History: Three Mile Island - March 28,1979
https://www.youtube.com/watch?v=SUct_69kNpQ

Why The Nuclear Accident Was Produced In The Chernobyl Plant Animation
https://www.youtube.com/watch?v=KIwpT-8RQbw
```

Sometimes you want a page with a menu link but no automatic title. Include a @title directive, but use page-level CSS to prevent displaying of the title on the page (but not in the menu):

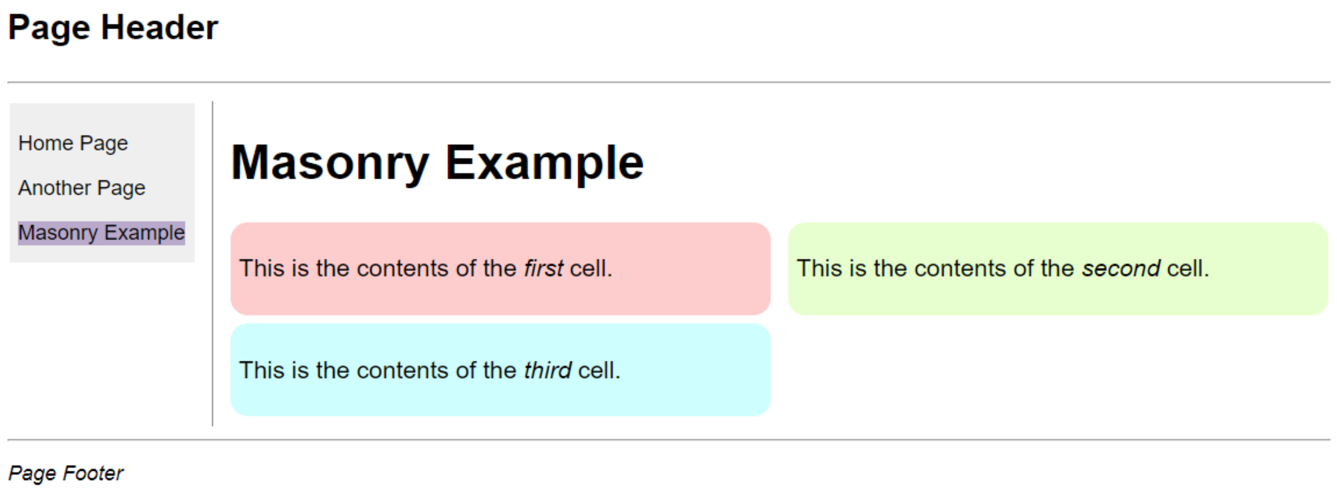
```
@title Another Page
<style>
	#title {
		display: none;
	}
</style>

Rest of page
```

If you want a page with an automatic title with the @title directives but which should not be in the menu, just edit the @menu file to remove its id. The id won’t be re-added when the site is rebuilt.

Here's an example of a page using Masonry:

```
@title Masonry Example
@masonry
@colors
%%cell
This is the contents of the *first* cell.
%%cell
This is the contents of the *second* cell.
%%cell
This is the contents of the *third* cell.
```

Normally Markdown formatting syntax isn't processed within block-level HTML tags, but you can use it inside Masonry cells, as shown. Figure 3 shows this page in a browser.



Masonry Example

This is the contents of the *first* cell.

This is the contents of the *second* cell.

This is the contents of the *third* cell.

Figure 3: Masonry page in browser

7. Final Words

SSB is a minimalist tool for building any static website at all. There are no limitations imposed by drag-and-drop systems, such as SquareSpace or Google Sites. It works well with AWS S3, and even a big site can be hosted for less than $10 a year (plus whatever your domain registration costs). It's a small Python program, available as open source, so you can easily understand what it's doing and change it to suit your needs.

From what I know, SSB is the fastest and simplest way to built a multipage website with menus.