Outline of *Working With Static Sites*

Bringing the Power of Simplicity to Modern Sites

Raymond Camden
Brian Rinaldo

Based upon *Working With Static Sites* by Raymond Camden & Brian Rinaldo (O'Reilly).

This document was:

Produced: 2019-01-05 11:34 Version: 0.0.7 Edition: 0.1

Table of Contents

Preface

Because of the **benefits** static site generators offer, static sites generators are used to run thousands of sites and are becoming the basis for a broad **set of tools** that reach the casual developer and the nontechnical content writer.

It can be difficult to know which tools to choose and how to get started.

That is the problem the authors hope to address in writing this book. By providing **common scenarios** and insights on how to address them, the authors hope to make it easier for anyone to create static sites solutions and take advantage of the speed, flexibilty, and security they offer.

What You Need to Know

Who this book is for

- This book is for web developers who are looking for a simpler way to build and deploy websites.
- For developers with experience with dynamic app servers (like PHP, Node.js, and ColdFusion), this book will present a simpler alternative.
- For developers who are still working with simple websites but need a way to make them more powerful.

What's not covered

- This book focuses on static site generators that work from the command line.
- Desktop tools that have similar features are not covered.

How this book is organized

- Begins by describing why you would want to use static sites.
- Subsequent chapters focus on a specific type of site and uses this as a way of introducing different static site generators.
- How to build a site
- More advanced topics, such as adding dynamic elements back in,
- Working with CMS
- How to deploy and host a site
- How to migrate from a dynamic site to a static one

Conventions

Italic Indicates new terms, URLs, email addresses, filenames, and file extensions

`Constant Width`
Used for program listings, as well as within paragraphs to refer to program elements such as variables or function names, databases, data types, environment variables, statements, and keywords

`*Command*`
: Shows commands or other text that should be typed literally by the user

‘`<Sample>`’
: Shows text that should be replaced with user-supplied values or by values determined by context.

Code Examples

Supplemental material (code examples, exercises, etc.) is available for download at `https://github.com/cfjedimaster/Static-Sites-Book`

1 Why Static Sites

Why would static sites be a worthwhile option for today's web? This chapter will explore some of the benefits of static sites before diving into how the changing technology behind static sites (i.e., static site generators—the topic of this book) are making them viable again.

1.1 Benefits of Static Sites

Of the many reasons that static sites are coming back into fashion, two stand out:

1. Static sites are *fast*
2. Statis sites are *secure*

1.2 Static Sites are *Fast*

Website performance is critical. Users tend to abandon sites that take longer than three seconds to load (with a load time of under two seconds being considered optimal for mobile). Achieving this level of website performance can be difficult.

By their very nature, static sites load extremely fast. This is because every visitor is served the exact same HTML without the bottlenecks caused by a server-side language, database, or any kind of dynamic rendering. Plus, static sites are extremely easy to cache and serve via a content delivery network (CDN), making them even faster for the end user. In addition, once you eliminate dynamic rendering from a database, you've eliminated numerous points of failure that often cause sites to be unresponsive or completely fail.

1.3 Static Sites are *Secure*

With a static site, there is no database to breach and no server-side platform or CMS with unpatched vulnerabilities. Static sites will not eliminate every vulnerability, but they narrow the window of opportunities available to any hacker and limit the amount of potential damage if a hacker does gain access.

1.4 Other Benefits

There are other benefits of static sites as well, including:

Flexibility
: You are not working within a CMS framework, so there are no limitations on how you can buidl your site.

Hosting
: Because there's no need for a database or server-side language support, hosting a static site can be anywhere from inexpensive to completely free, depending on your needs.

Versioning
: Since a static site is made up of static files, it is extremely easy to track and coordinate changes using version control systems like Git and GitHub.

With all of these benefits, why wouldn't you choose to use a static site? Well, only certain kinds of sites can realisticly work as static only.

1.5 What Kind of Sites Can Go Static

There are drawbacks to using static sites.

- While some amount of dynamic data is possible on a static site that uses external API calls or third-party services, a static site is simply not suitable if you require large amounts of dynamic data or content personalization.
- From a development and content contribution standpoint, static site generators (i.e., the tools frequently used to build static sites) can have a steep learning curve.
- Deployment can be complex, making static sites less than ideal for content that changes frequently.

Sites that tend to work best as static sites are content-focused, infrequently updated (once or twice a day at most), and do not require a high degree of user interaction or personalization.

Here are some examples of types of sites that work well as static sites:

Blogs
: This is the most common use case; many static site generators default to a blog template. Blogs are content-focused by design and, in many cases, user interaction is limited to comments, where services like Disqus (`https://disqus.com/`) con fill the requirement.

1.6 What Are Static Sites

Static site generators solve the pain of building and maintaining a static site. The fundamentals of a static site generator are extremely simple: they take in dynamic content and layout and output static HTML, CSS, and JavaScript files. There are literally hundreds of static site generators (`https://staticsitegenerators.net/`) but essentially they all do exactly the same thing and, for the most part, function similarly.

- lightweight markup language (`Markdown .md`)
- a templating language (`Handlebars {{}}`)
- data/metadata (`Yaml .yml`)
- CSS preprocessor (`Sass .scss`)
- a compile-to-JavaScript language (`CoffeScript .coffee`)
- ⇓
- S S G ⇒ `.html .css .js`

What SSG's Have In Common

Templating language
: Use one or more, such as Liquid, Handlebars, Jade. This is a key part of any static site generator, as it allows one to build a layout/theme for a site and plug in dynamic content during a build.

Markup language
: Use one or more (`http://bit.ly/2lrz8JP`) such as Markdown, AsciiDoc, reStructuredText. Using a lightweight markup language makes it quicker and easier to write content using any text editor.

Command line
: Are run via the command line.

Local development server
: This allows you to develop and test locally befdore building and deploying your changes. Typically, the tool watches the folder where the site files are being edited and recompiles the site on-the-fly as you edit them.

Extensible
: In most cases, if your static site generator doesn't support a feature or language that you need, it has a build-in plugin architecture that allows you to add that in (provided you can code in the language the tool is built upon).

Support file-based data formats
: e.g. YAML, TOML, JSON. File-based data allows you to structure any type of data independent of its display.

The author has published a report called *Static Site Generators: Modern Tools for Static Web Development (O'Reilly 2015)*. The report focuses on the history of static sites, how they differ from dynamic sites built using content management systems or blog engines, and gives some details about the available static site generators.

The report is free and can be downloaded from O'Reilly here: `http://oreil.ly/2l4fL8j`.

1.6.1 How to Choose a Static Site Generator

So now with a basic understanding of what a static site generator is, how can one use them? The first issue to resolve is trying to figure out which one to use. This isn't an easy decision, since there are in excess of 445 different available options. ([List of SSG's], page 4). Even after filtering out projects that haven't been updated recently, there are still hundreds of potential tools.

How To Choose the Right SSG?

The relative "health" of the project
: How recently has it been updated, and how big and active is the community?

How good is the documentation
: Many of the hundreds of static site generators (probably a majority) are not well documented. This can cause you to spend far more time than necessary building your site and lead to needless frustration. Review the documentation carefully before committing to a project and don't assume that being able to read the source is sufficient.

Does it support my requirements
: This can involve very specific requirements (an importer or specific plugin) or a more general ability to meet amy needs (is it even extenible?).

Is the language it is built upon important to me
: For most, a static site generator will work well out of the box. But in some cases it might be necessary to customize your generator via plugins or even contributions to the source code. In these cases, knowing the underlying language can be important.

1.6.2 Going Forward

The next chapters will look at some of the more mature and popular options for developing static sites, such as:

- Jekyll (Ruby)
- Hugo (Go)
- Harp (JavaScript)

Not only does each have a different underlying language, which may be an important consideration, but each also has its own pros and cons. The chapters will look at building some common use cases:

- a basic informational site
- a blog
- documentation site

using these tools in ways that take advantage of their relative merits.

Dynamic Features

Once the basics of the static site have been built, there needs to be added some dynamic features, like comments on a blog post, or a site search. Or, if we have to support content contributors that aren't comfortable writing posts in Markdown via a text editor, we might want to add a CMS-like backend to the site to allow for easy editing. The following will take a look at multiple solutions that solve each of these problems.

Deployment

After the site is complete, it's time to deploy it. While this can be as easy as simply FTPing files onto a server, in most cases you'll want to automate the process or take advantage of services that can manage the build process for you. The following chapters will explore a variety of tools and services that can ease the deployment process.

Migration

Finally, you may be evaluating static sites as an option to replace an existing site that uses a tool like WorkPress or some other CMS. For these cases, the chapters will dive into tools that ease the process of migrating to a static site generator from a CMS by bringing over existing content.

SSG Ecosystem

In the end, this book hopes to provide you with a broad overview of the existing statit site generator ecosystem, while diving into the actual implementation details of how to accomplish your goals with these tools.

2 Building a Basic Static Site

2.1 Welcome to Harp

2.2 Your First Harp Project

2.3 Working With Layouts and Partials

2.4 Working With Data

2.5 Generating a Site

2.6 Building Camden Grounds

2.7 Going Further With Harp

3 Building a Blog

4 Building a Documentation Site

4.1 Characteristics of a Documentation Site

4.2 Choosing a Generator for Your Documentation Site

4.3 Our Sample Documentation Site

4.4 Creating the Site

4.4.1 Installing Hugo

4.4.2 Generating the Initial Site Files

4.4.3 Configuring the Hugo Site

4.4.4 Adding Content

4.4.5 Creating the Layout

4.5 Going Further

5 Adding Dynamic Elements

5.1 Handling Forms

5.1.1 Wufoo Forms

5.1.2 Google Docs Forms

5.1.3 Formspree

5.1.4 Adding a Comment Form to Camden Grounds

5.2 Adding Comments

5.2.1 Working with Disqus

5.2.2 Adding Comments to The Cat Blog

5.3 Adding Search

5.3.1 Creating a Custom Search Engine

5.3.2 Adding a Custom Search Engine to a Real Site

5.4 Even More Options

6 Adding a CMS

6.1 CloudCannon

6.1.1 Creating a Site on CloudCannon

6.1.2 Editing a Site on CloudCannon

6.1.3 Where to Go from Here

6.2 Netlify CMS

6.2.1 Setting Up the Netlify CMS

6.2.2 Where to Go from Here

6.3 Jekyll Admin

6.3.1 Setting Up Jekyll Admin

6.3.2 Editing a Site in Jekyll Admin

6.3.3 Where to Go from Here

6.4 More Options

6.4.1 Forestry.io

6.4.2 Lektor

6.4.3 Headless CMS

7 Deployment

7.1 Plain Old Web Servers

7.2 Cloud File Storage Providers

7.2.1 Hosting a Site on Amazon S3

7.2.2 Hosting a Site on Google Cloud Storage

7.3 Deploying with Surge

7.4 Deploying with Netlify

7.5 Summary

8 Mirgrating to a Static Site

8.1 Migrating from WordPress to Jekyll

8.2 Other Migration Options

8.2.1 Hugo

8.2.2 Middleman

8.2.3 Hexo

8.2.4 Harp

8.2.5 Many More Options Are Available

8.3 Go Forth and Be Static

Index