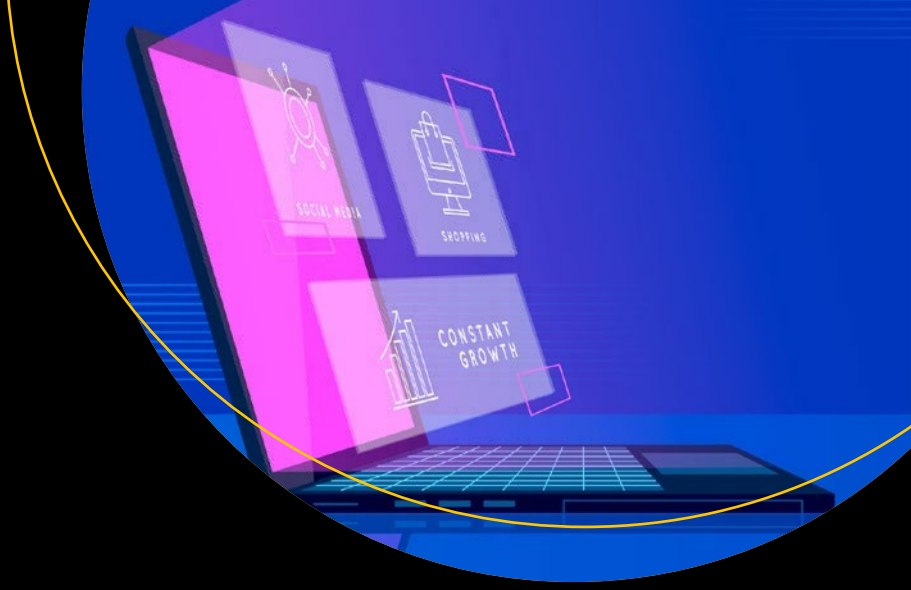

Using Gatsby and Netlify CMS

Build Blazing Fast JAMstack Apps Using Gatsby and Netlify CMS

—

Joe Attardi

Apress®

Using Gatsby and Netlify CMS

Build Blazing Fast JAMstack Apps Using Gatsby and Netlify CMS

Joe Attardi

Apress®

Using Gatsby and Netlify CMS: Build Blazing Fast JAMstack Apps Using Gatsby and Netlify CMS

Joe Attardi
Billerica, MA, USA

ISBN-13 (pbk): 978-1-4842-6296-2 ISBN-13 (electronic): 978-1-4842-6297-9
https://doi.org/10.1007/978-1-4842-6297-9

Managing Director, Apress Media LLC: Welmoed Spahr
Acquisitions Editor: Louise Corrigan
Development Editor: James Markham
Coordinating Editor: Nancy Chen

Cover designed by eStudioCalamar

Cover image designed by Freepik (www.freepik.com)

Distributed to the book trade worldwide by Springer Science+Business Media New York, 1 New York Plaza, New York, NY 10004. Phone 1-800-SPRINGER, fax (201) 348-4505, e-mail orders-ny@springer-sbm.com, or visit www.springeronline.com. Apress Media, LLC is a California LLC and the sole member (owner) is Springer Science + Business Media Finance Inc (SSBM Finance Inc). SSBM Finance Inc is a Delaware corporation.

For information on translations, please e-mail booktranslations@springernature.com; for reprint, paperback, or audio rights, please e-mail bookpermissions@springernature.com.

Apress titles may be purchased in bulk for academic, corporate, or promotional use. eBook versions and licenses are also available for most titles. For more information, reference our Print and eBook Bulk Sales web page at http://www.apress.com/bulk-sales.

Any source code or other supplementary material referenced by the author in this book is available to readers on GitHub via the book's product page, located at www.apress.com/9781484262962. For more detailed information, please visit http://www.apress.com/source-code.

Printed on acid-free paper

To Liz and Benjamin – you are my whole world.

Table of Contents

About the Author

Joe Attardi is a software engineer specializing in front-end development. He has over 15 years' experience working with JavaScript, HTML, and CSS and has worked extensively with front-end technologies such as Angular and React. He currently works at Salesforce and has worked in the past with companies such as Dell and Nortel. He is also the author of *Modern CSS*, an Apress title. He lives in the Boston area with his wife and son. You can find him on Twitter at @JoeAttardi.

About the Technical Reviewer

Alexander Nnakwue has a background in Mechanical Engineering from the University of Ibadan, Nigeria, and has been a front-end developer for over 3 years working on both web and mobile technologies. He also has experience as a technical author, writer, and reviewer. He enjoys programming for the Web, and occasionally, you can also find him playing soccer. He was born in Benin City and is currently based in Lagos, Nigeria.

Acknowledgments

I'd like to start by thanking my wonderful wife, Liz, for her constant love and encouragement throughout the whole writing process – and for understanding when I locked myself away in solitude to write. And my little toddler, Benjamin, for giving me much needed breaks from writing for play time.

Thanks to all my friends and family for always supporting and encouraging my interest in computers and technology.

This book began its life as a self-published work, and I'd like to thank Apress for making it what it is today. I'd also like to thank the awesome team at Apress – Louise Corrigan, Nancy Chen, and Jim Markham – for guiding me through the process every step of the way. I appreciate their patience with me as a first-time author.

Thanks to Alexander Nnakwue, the technical reviewer for this book, for his time and excellent feedback, helping make this book even better.

Introduction

In this book, you will learn all about creating a website using Gatsby, getting its content from Netlify CMS, a free content management system from Netlify. We'll start with a bare-bones template project and install and configure Netlify CMS from scratch.

In Chapter 1, we'll look at the JAMstack and what a content management system is. We'll talk about headless CMSs and introduce Netlify CMS.

Chapter 2 is a Gatsby crash course. If you have built a site with Gatsby before, you can probably skip this chapter, as it is a very basic overview of Gatsby. Conversely, if you haven't been exposed to Gatsby before, this introduction should be enough to get you going.

We'll start setting up the example project, a coffee shop website, in Chapter 3. We'll take care of signing up for GitHub and Netlify if you haven't already and also get your version of the example project deployed on Netlify.

In Chapter 4, we'll start adding Netlify CMS to the project. This covers installation of the CMS and creation of the configuration file. We'll set up the blog content and integrate with the Netlify identity service.

In Chapter 5, we'll take a tour of the CMS user interface, and create our first piece of content: a blog post. We'll also look at how to add images to blog posts.

We'll take the content created by the CMS and start integrating that with Gatsby in Chapter 6. We'll configure some Gatsby plugins which will add the blog content to Gatsby's data model.

Chapter 7 is all about creating pages from data. It will introduce the Gatsby Node APIs and show how Gatsby dynamically creates pages by combining data with a page template.

Eventually, we'll end up with too much blog data to show on a single page, so in Chapter 8, we'll learn how to paginate the blog posts. We'll split the blog post listing into several pages (which are also dynamically created).

Up to now, the content on the landing page, such as the hero image and tagline, is managed in the source code. Chapter 9 will explore how to make this content dynamic and editable from Netlify CMS. We'll make several elements of the landing page configurable from the CMS, allowing the page content to be updated without writing a line of code.

In Chapter 10, we'll create a menu for the coffee shop. We'll create new content definitions in the CMS configuration for menu categories, items, and prices.

Gatsby has some plugins that can help generate more efficient images to improve page load time. We'll look at how to do this in Chapter 11.

In Chapter 12, we'll see how we can customize the CMS user interface, extending it with a custom preview component so that we can see a live preview of how the menu looks from within the CMS.

Finally, in Chapter 13, we'll look at how to leverage Netlify CMS's editorial workflow, creating draft content and reviewing it via GitHub pull requests.

We'll close the book in Chapter 14 with some discussion of Netlify CMS beta features and pointers to more Netlify CMS resources.

CHAPTER 1

Introduction to Netlify CMS

Before we dive into the specifics about Netlify CMS, let's look at a few introductory topics. These include the JAMstack, the Netlify service, and content management systems in general. From there, we'll start looking at Netlify CMS itself.

The JAMstack

Traditional web application stacks such as LAMP (Linux, Apache, MySQL, PHP) or MEAN (MongoDB, Express, Angular, Node.js) specify particular server platforms, programming languages, and database vendors.

JAMstack - that's JavaScript, APIs, and Markup - is a new paradigm for building fast web applications. It is technology-and tool-agnostic. A JAMstack site might be powered by JavaScript via Gatsby or by Ruby via Jekyll.

A JAMstack site pre-renders its pages to static HTML markup which can be served from a CDN or other hosting service. These pages contain JavaScript to provide interactivity, which loads data from an API of some kind. In this model, the application is decoupled from its data.

The API could be a backend service deployed separately, or it could be a third-party service like a headless CMS or a set of AWS Lambda functions.

Figure 1-1 shows an overview of the JAMstack architecture.

J. Attardi, *Using Gatsby and Netlify CMS*, https://doi.org/10.1007/978-1-4842-6297-9_1

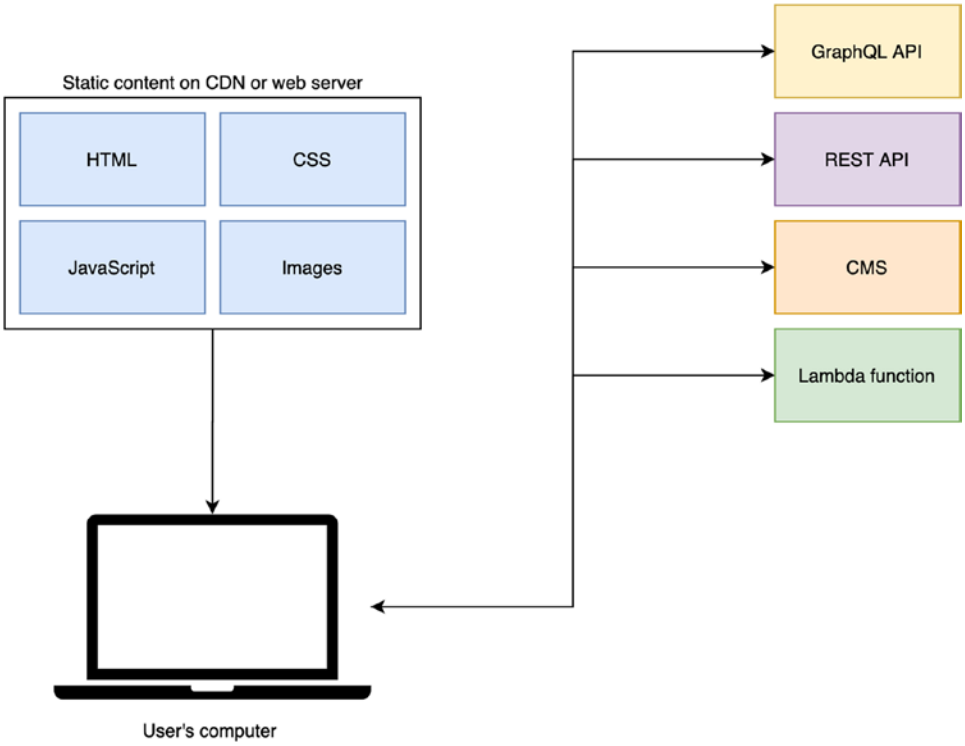

Figure 1-1. *The JAMstack*

One of the biggest benefits of using the JAMstack is performance. Because the pages are generated ahead of time and deployed as static HTML, there is no dynamic generation of HTML from a server and no wait for server-side code to execute. Instead, these static resources can be served from a lightning-fast content delivery network (CDN). By using a CDN, assets are served to a visitor from a server geographically close to them.

There can also be cost savings. Serving your static assets from a CDN is likely less expensive than maintaining your own server infrastructure.

About Netlify

Netlify (`https://netlify.com`) is a popular platform for hosting websites and applications. At its core, it is a hosting service but also has many other powerful features such as analytics, serverless functions, forms, and a CDN. They have a generous free tier of services.

Netlify's workflow is based on Git repositories. A site is created by connecting a repository from a service like GitHub or GitLab. When a new commit is pushed, Netlify will build and deploy a new version of the site. For this reason, the Netlify platform is well suited to building JAMstack applications.

In this book, we will build and deploy the example project on the Netlify platform.

What is a CMS?

As a web developer, it is easy to build web content by just writing HTML and CSS (or, in the case of Gatsby as we'll see later, Markdown). But consider a nonprofit organization that hires a developer to build a website for them. Unless someone at the nonprofit knows how to code, they will have to spend more money by having the developer come back every time they need the site updated.

A content management system, or CMS, is a solution to this problem. Instead of the content being written in source code, the CMS provides a rich editing experience for authoring and maintaining content. Some features of a CMS might include a WYSIWYG (What You See Is What You Get) editor or a media library for managing images and other media.

A CMS allows multiple authors to draft, edit, and publish content to the site, all without having to worry about writing markup or code. It's also easier because content can be edited directly in the web browser.

A CMS can be used to manage multiple types of content on a site. The most common type of content that comes to mind is an article or blog post. However, a CMS can even be used to provide content on a landing page, header, or footer as well.

Traditional CMS

A traditional CMS is a large web application that handles both the management and display of the content. The content is typically stored in a database of some kind. An author logs in via a web interface, edits the content, and it is saved to the database.

A visitor to the site will utilize the same application to view the content. It retrieves the content from the database where it is presented to the user.

Figure 1-2 shows an overview of a traditional CMS platform.

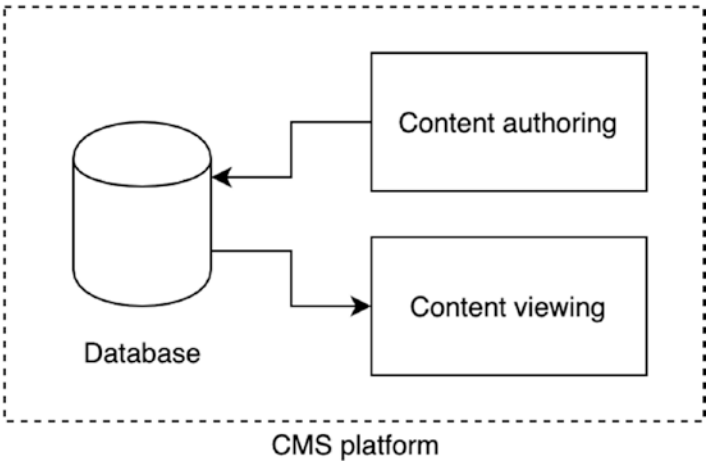

Figure 1-2. *The architecture of a traditional CMS*

Some examples of traditional CMS platforms are WordPress, Drupal, and Joomla.

Headless CMS

A headless CMS is a platform for authoring content but does not contain a front-end interface for displaying that content. Instead, the content is exposed through an API of some kind.

The primary benefit of a headless CMS is that since the content is delivered via an API, it can be consumed by multiple front-end applications. It also provides greater flexibility in how the content is presented. A traditional CMS may be highly customizable, but you are still locked into the platform.

Figure 1-3 shows a typical headless CMS.

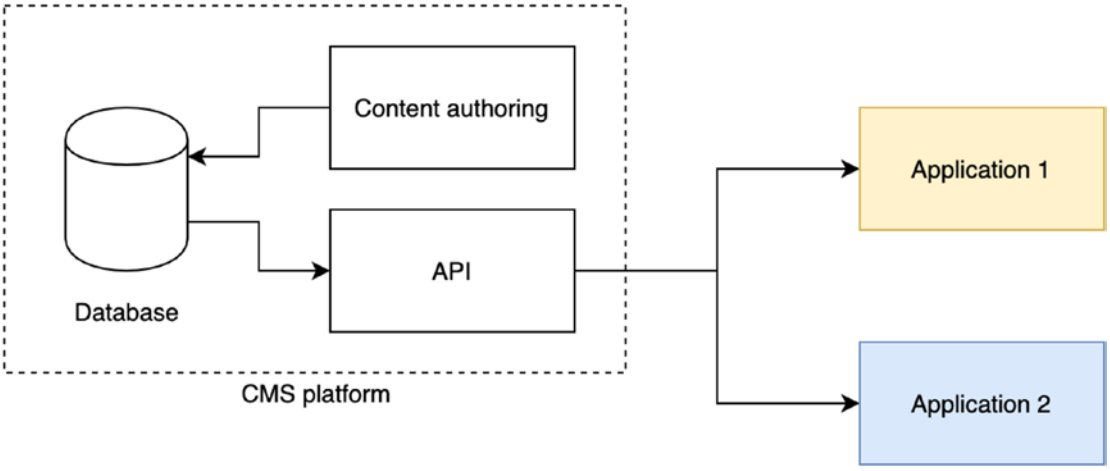

Figure 1-3. *The architecture of a headless CMS platform*

Some examples of headless CMS platforms are Contentful, Magnolia, and Netlify CMS.

As you can probably tell, a headless CMS is a perfect choice for a JAMstack application due to its decoupled, API-first nature. We can build an entire site around the API, using it as a data source for our content.

One potential drawback of a headless CMS is that you may not be able to see a preview of how the content will look on the live site. Fortunately, Netlify CMS supports custom previews, which solves this particular problem.

Netlify CMS

Netlify CMS is a free, open source headless CMS created by Netlify. Despite its name, it is not locked to the Netlify platform. While it is much easier to build and deploy a Netlify CMS-powered site on Netlify, it is not required.

Some CMS platforms store their data in databases that are accessed via an API. Netlify CMS takes a simpler approach. The content is stored as images and Markdown files in a Git repository. This makes it an ideal partner with Gatsby, which has plugins for working directly with, and rendering, Markdown data. It should be noted that Netlify CMS works with other site generators and platforms as well.

Netlify CMS provides a React single-page application with a rich text editor so that content creators don't have to write Markdown directly. While the user interface is powered by React, it can be used with a site built with any technology. Netlify CMS can be used with any platform, such as Hugo, Jekyll, Nuxt, as well as Gatsby.

Backends

Netlify CMS supports several different backends for storing its data. These different backends also provide authentication mechanisms for logging into the CMS. Some of these backends include GitHub, GitLab, and Bitbucket. For these three backends, a user can log in using their GitHub, GitLab, or Bitbucket account, respectively.

Another backend (and the one we will use in this book) is the Git Gateway backend. This is a Netlify-provided backend that does not require the credentials to the user's Git provider. Instead, authentication is configured via the Netlify Identity service, which includes a basic application for managing users. At the time of writing, the Git Gateway backend is only supported for GitHub or GitLab repositories.

The Git Gateway backend is deployed with your site on Netlify under a directory called `.netlify` which contains a basic set of APIs for creating the Git commits. When new content is added, the browser calls these APIs which have some server-side code which will communicate with the configured Git provider's API to create the new Git commit.