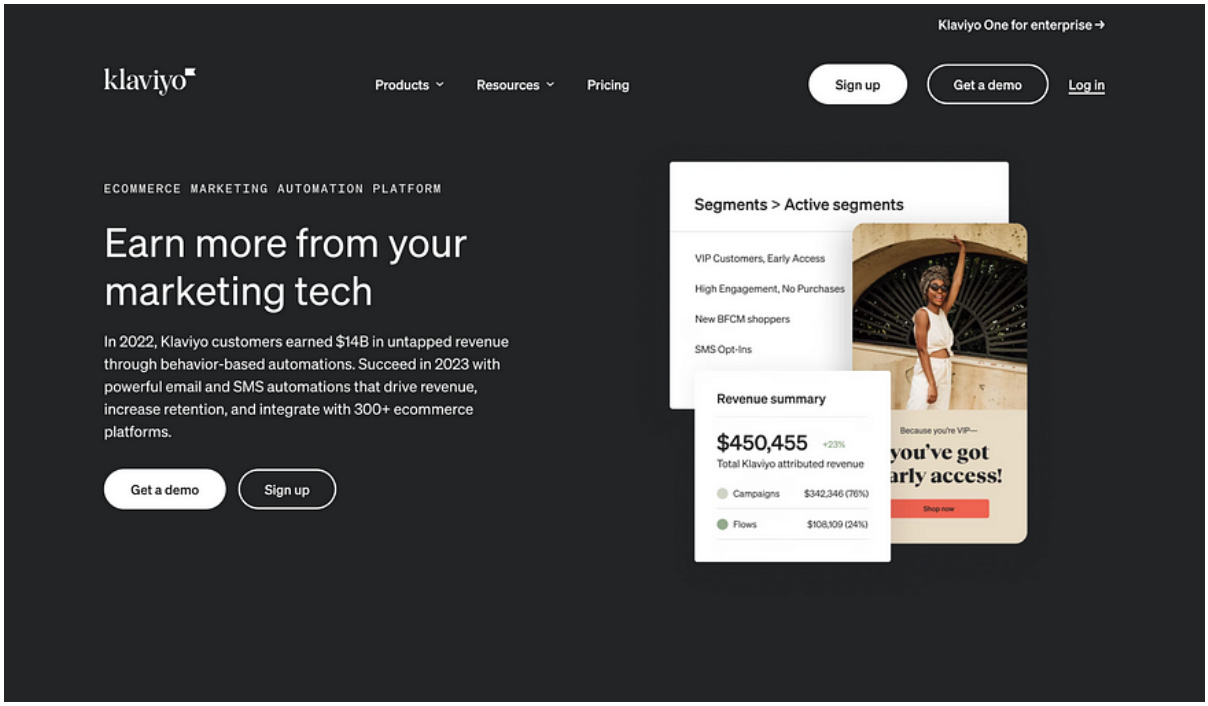
How we built Klaviyoâ€™s new marketing site

Author: Jon Darby

Claps: 73

Date: Mar 28

About a year ago, Klaviyo launched a completely rebuilt marketing site. Hereâ€™s how we did it.



Where we started

Our previous marketing site, which from here on out Iâ€™m going to call the â€œlegacy site,â€ was a fairly traditional WordPress setup. Its core technologies will be familiar to folks in the WordPress community:

- Roots Sage, including Laravel Blade templates
- Advanced Custom Fields for structured data
- Gutenberg for newer blog posts

The legacy site featured a fully containerized architecture hosted on Pantheon, a deployment pipeline powered by GitHub Actions (including an isolated preview environment for each pull request), and visual regression testing via BackstopJS. These features produced a great developer experience and made it easy to ship quickly and confidently.

Like any years-old content management system (CMS), ours was full of content, media items, and plugins. A quick peek at the archive revealed more than 12,858 media items and 1,984 posts

and pages. Before my team took it on, the system didnâ€™t have a consistent technical owner, so its codebase was full of solutions that were purpose-built for a single campaign or page, leading to a maze of templates, functions, and styles. These produced performance issues and increased page load times, negatively impacting the visitor experience.

Planning a new site

Klaviyoâ€™s rebrand offered the perfect opportunity to rebuild the marketing site from the ground up and address the shortcomings of the legacy site. When considering approaches, we kept in mind our two primary audiences: external visitors and internal publishers. We settled on the following areas of focus for evaluating potential solutions:

- **Performance**
  The new site needed to be fast, as measured by page load times and Lighthouse metrics
- **Search Engine Optimization**
  It needed to be discoverable and rank well in search results for branded and non-branded keywords
- **Modern frameworks, tools, and approaches**
  We needed to build with best-in-class technologies that were aligned with the rest of the organization and industry, making it easier to attract talent, and increasing the ability of talent to work across projects at Klaviyo
- **Familiar and delightful editor experience**
  Stakeholders should experience minimal interruption to their content publishing workflows, and their publishing experience should be fast, simple, and enjoyable
- **Speed of delivery**
  We needed to ship the new site on a tight timeline

We developed a spreadsheet to help us evaluate different solutions, assigning points based on the criteria above. To spare you the gory details, hereâ€™s where we landed:

- **React**
  React is used widely at Klaviyo and is an industry-leading framework with a robust ecosystem.
- **Gatsby**
  For search engine optimization and performance, we needed to deliver pre-rendered HTML over the wire. We considered several static-site generators, narrowing to Next.js and Gatsby. We ultimately chose Gatsby due to its tight CMS integrations, image handling (our site is media-rich), and our teamâ€™s past experience using it.
- **WordPress**
  After evaluating multiple headless CMS solutions, we decided to retain WordPress as our content management system. This preserved a familiar editing experience for our contributors and minimized the amount of technical change on the project, a feature that allowed us to get up and running quickly.

With those high-level selections made, we had a solid foundation on which to build. We needed to deliver an entirely new site on an aggressive timeline, so we worked with our partners in marketing to identify a subset of pages that could serve as the core of the new site.

We landed on a plan to maintain the legacy site (with a fresh coat of paint) and the new site side-by-side for a limited amount of time, enabling us to focus engineering efforts on launching a subset of rebranded pages, while allowing marketing and content teams additional time to migrate legacy content to the new site.

This plan, and our technical selections, were validated by engineering stakeholders through Klaviyoâ€™s Request for Comment (RFC) process, through which our team gained valuable insight and feedback.

Starting the build

As part of the rebrand, Klaviyoâ€™s creative team created a design system in Figma for the brandâ€™s identity and marketing products. Included in the system was a set of components for web properties, which could be combined to create visually coherent yet unique page layouts.

This bespoke design system meant using an opinionated component library or styling framework was off the table, as doing so would mean stripping down much of the built-in styling, and building ours on top. For this reason, we chose to build the design system from scratch using styled-components and Storybook for documentation, both of which were already in use by other teams at Klaviyo. We adopted an atomic design approach, organizing the components into atoms, molecules, and organisms.

Configuring the CMS

To make WordPress as performant as possible, we decided to stand up an entirely new installation. A blank slate meant we could iterate quickly and experiment with different data structures and organizational schemes within the CMS without disrupting existing content workflows. Further, it meant we wouldnâ€™t be dragged down by the thousands of pieces of legacy content, and allowed us to focus our limited time on building a forward-looking solution, not sorting through dead code.

Gatsby has a â€œbatteries-includedâ€ solution for WordPress: WP Gatsby. This, along with WP GraphQL and Advanced Custom Fields (ACF), formed the foundation of our new CMS. Using ACF, we created Field Groups aligned to our components in the atomic design system and used those Field Groups in template-level Flexible Content layouts. This setup gave editors control over the components and their order on a page while providing a simple, predictable data structure for engineers to consume.

WP Gatsby and WP GraphQL together gave us a number of features that saved significant engineering effort, including:

- A GraphQL API endpoint populated with fields built into WordPress and those added via Advanced Custom Fields
- Visual previews of WordPress content via Gatsby Cloudâ€™s CMS Preview functionality
- Support for incremental builds, wherein Gatsby builds only content that has changed since the last build, decreasing build and publishing times

Building with Gatsby

With a functional content management system in place and preconfigured for Gatsby, setting up the Gatsby site was relatively simple. We began with a minimal Gatsby WordPress starter â€” essentially, a barebones Gatsby site ready for customization. This provided an organizational structure for the Gatsby site and preconfigured essential plugins, like gatsby-source-wordpress and gatsby-plugin-image, with solid defaults.

The majority of our engineering effort was invested in building custom components. We made an intentional decision to develop a robust set of components populated by CMS-provided data rather than hand-coding individual pages or layouts. This required our team to invest more effort up front, but made the engineering effort required to stand up new pages near-zero. That decision paid off when the creative and marketing teams began entering content in WordPress in earnest. Our gorgeous site materialized practically overnight!

This left us with one last task: sharing it with the world.

Deploying the site

Klaviyo uses NGINX as a reverse proxy to serve all of www.klaviyo.com, a subdomain shared by the marketing site and the Klaviyo SaaS application. This gave our team flexibility to gradually cut over traffic to the new site on launch day and the ability to select an upstream (legacy site or new site) on a per-page basis.

We used weighted load balancing with sticky sessions to gradually increase the amount of traffic routed to the new site. We began with an early-morning canary deployment with a single-digit percentage of traffic served by the new site, and closely monitored Sentry, Google Analytics conversion metrics, and Splunk for signs of regression.

Over the course of launch day, we increased the weight of the new site upstream, ultimately routing all traffic there by the end of the workday.

Maturing the solution

Our work didnâ€™t stop after deployment. Over the past year, weâ€™ve been hard at work adding features and maturing the solution. Hereâ€™s just a taste of what weâ€™ve been up to:

- Adding new components to the design system
- Sunsetting the legacy site
- Adding a subsite for our enterprise offering
- Converting the codebase to TypeScript
- Creating custom GraphQL types to improve the reusability of our components in Advanced Custom Fields
- Adopting a monorepo structure to ease development of additional sites
- Further modularizing our configuration through Gatsby plugins and themes

I look forward to sharing more about these efforts in future posts.

If youâ€™re interested in working on projects like this, my team (K-Ops Marketing) is hiring! Check out our openings on Greenhouse. Thanks for reading!