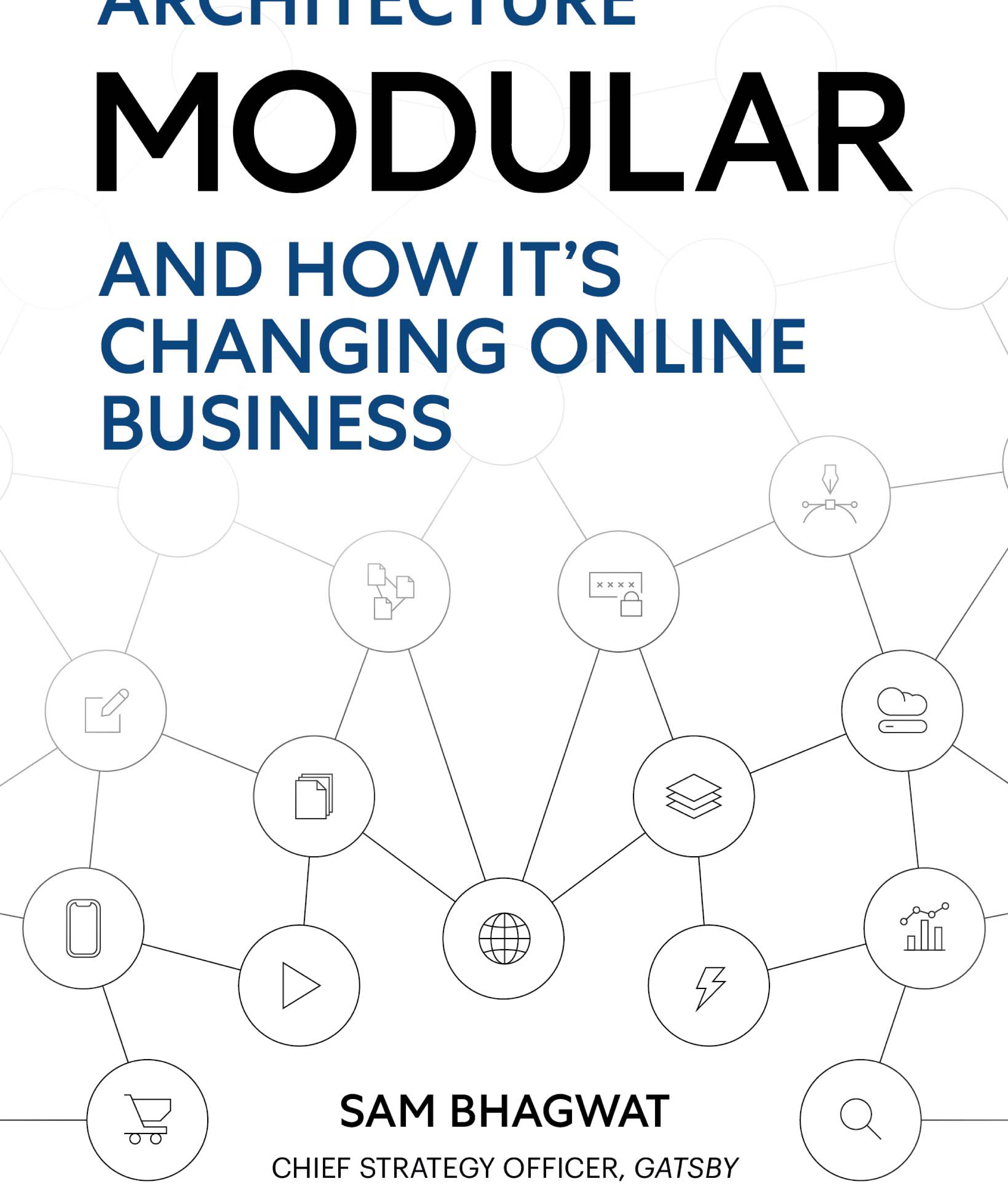
FOREWORD BY SASCHA KONIETZKE, *CONTENTFUL*

THE WEB'S NEW ARCHITECTURE MODULAR AND HOW IT'S CHANGING ONLINE BUSINESS



SAM BHAGWAT

CHIEF STRATEGY OFFICER, *GATSBY*

MODULAR

THE WEB'S NEW ARCHITECTURE
AND HOW IT'S CHANGING
ONLINE BUSINESS

SAM BHAGWAT

First Printing, 2022

Edited by Gary Zhexi Zhang
Cover image and design by Vlad Afanasiev
Typeset in Centra No.1 and Graphik

ISBN 9798352203651

Gatsby
548 Market St
#36791
San Francisco
CA 94104

www.gatsbyjs.com

MODULAR

THE WEB'S NEW ARCHITECTURE
AND HOW IT'S CHANGING
ONLINE BUSINESS

CONTENTS

FOREWORD

Sascha Konietzke
Co-Founder and Chief Strategy Officer
Contentful

A lot has changed in the past decade. Our expectations as consumers. How we work. The speed and adaptability that businesses need to be successful. Although many of these changes were already happening, the COVID pandemic accelerated them and made sure that there's no going back. The relationship between business and technology has fundamentally shifted toward a new model and a different way of thinking, one that increasingly dictates who'll win and why. And that new model is Modular, the subject of this book, and what Sam and myself believe is the future.

Back in 2010, I was working on several iPhone app projects. I quickly saw that existing Content Management Systems (CMSs) were great for enabling marketers to manage static web pages, but these CMSs weren't built to deliver interactive apps.

Frustration with this approach led me to co-found Contentful in 2012, the first cloud-based, headless CMS. It had no built-in presentation layer (the head), but instead focused on delivering content through a powerful API. The early Contentful customers were largely iOS and Android developers, working on native mobile apps. Early on, they saw that headless was a better way to develop, and they soon wanted to use headless for building websites as well. But there was a snag: no standard approach for creating a head for a headless CMS. The web presentation layer had to be custom-made.

That's why I was excited when I first met Sam and Kyle, the founders of Gatsby, in 2017 in San Francisco, for lunch. With Gatsby, they created a modern, decoupled presentation layer—the missing puzzle piece for a more flexible, modular web stack. We realized that we not only shared the same frustrations with the existing monolithic web CMS but had a shared vision of what it takes to build a better modular architecture as an alternative.

We weren't the only ones to share this vision of an alternative architecture. Impossible Foods, a leader in plant-based proteins, was experiencing exponential growth, but their tech stack couldn't keep up. Their monolithic CMS would only support static images; developer turnaround times were long and couldn't keep up with the requests. To support their growth, they needed an accessible, adaptable, and modular architecture. With Contentful and Gatsby in their stack, content creators got an inventory of modular page options, developers could focus on strategic work, and users visited a site that was now fast, beautiful, and dynamic.

The COVID pandemic forced companies, at any stage in their digital transformation journey, to build digital products even faster to reach customers. The only way to win was with composable content and a modular architecture, stacking components together as needed, and iterating quickly based on customer feedback.

At Contentful, we saw a great example of this. Seeing an increase in online sales at the start of 2020, IKEA wanted to shift to a multichannel, content-rich digital experience. To do that, they needed to take down the content silos that existed and find a solution to deliver adaptive content for various channels and regions. Now, all that lagom-home-decor inspiration is digital and modular. In the wake of the pandemic, this proved to be a prescient investment.

Ten years ago, the modular ecosystem was a small group of developers. Today, it's a movement that includes some of the largest companies in the world, and includes a mature ecosystem with dozens of battle-tested solutions.

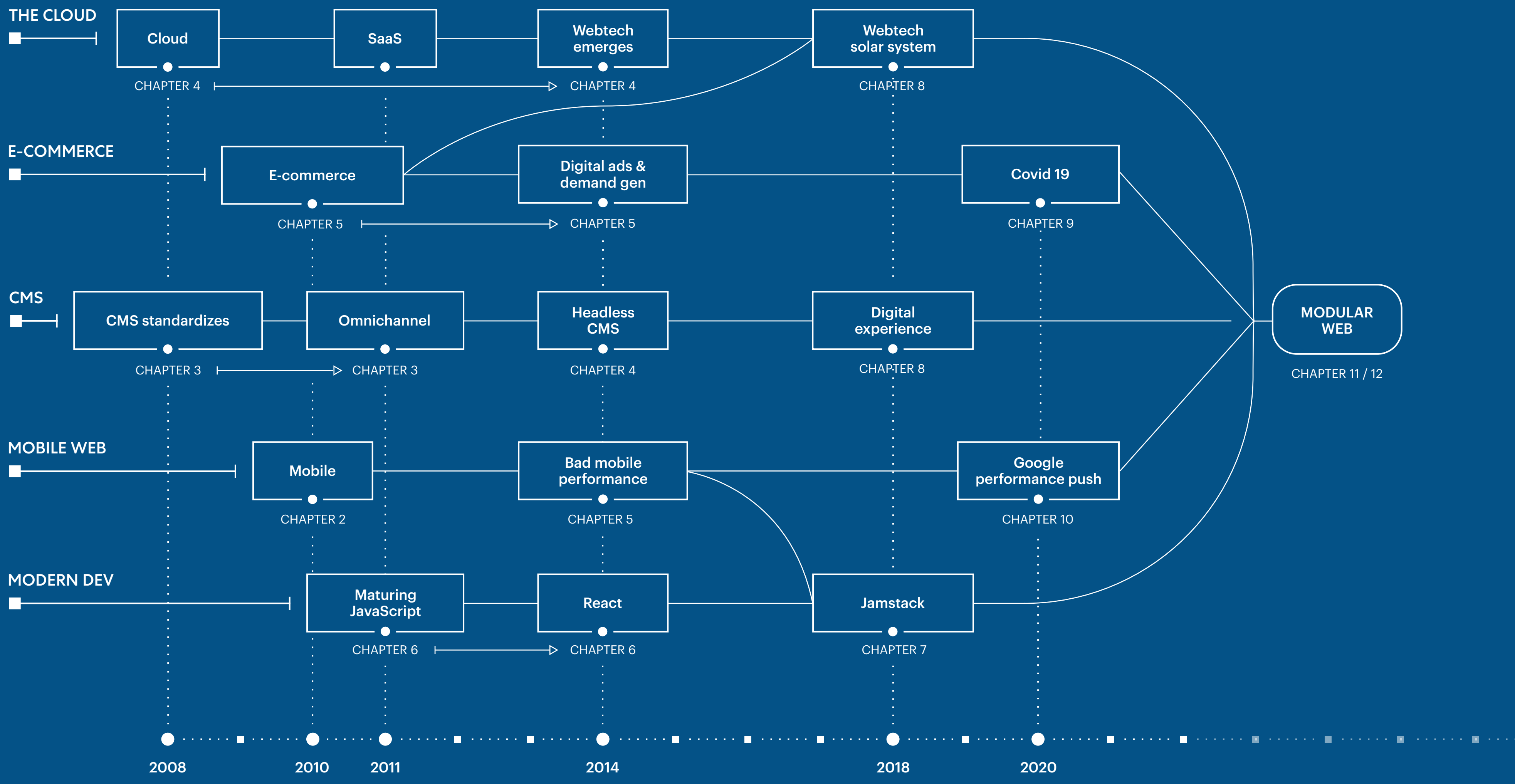
Understanding modular architectures and the rapidly evolving ecosystem around it matters to more than just developers—it is a requirement for anyone whose job involves content or customer experience. In this book, Sam provides historical knowledge and understanding of the modular ecosystem and why it is the way forward.

When I started Contentful, there was no manual for how to build better for the modern web. I'm really excited that Sam has now written the missing manual.

Sascha Konietzke
Berlin, August 2022

TIMELINE

RISE OF THE MODULAR WEB



INTRODUCTION

CHAPTER 1

HOW WE BUILD WEBSITES

WHY YOUR WEBSITE MATTERS • WHAT THE MODULAR WEB IS • WHY READ THIS BOOK

The last decade has turned the web upside down.
The web used to be primarily used on desktops. Today, most traffic happens on mobile.
Buying products online used to be occasional. Now, it's routine.
As customers spend more time online, every consumer business is becoming an online business.

Digital ad spend is skyrocketing, and marketing departments are building data-driven "demand generation" teams to optimize spend and conversion.

In the 2010s, the best bet for most businesses who cared about their website was a great content management system (CMS). Good content workflows and content structure created agility. Productive teams could quickly launch new campaigns and optimize key conversion funnels.

In the 2020s, increasing standards and greater competition is pushing teams toward the "modular web." Cloud-based CMSs that foster intuitive content modeling. Development frameworks that maximize website performance and use modern JavaScript libraries. Marketing analytics tools that give maximum granular visibility. Design that allows for 3D visuals and effects.

Teams are turning to cloud-based, SaaS tools for functionality like search (auth, chat..). They are combining API-based "headless" CMSs with a modern JavaScript development environment to build high-performing websites.

The goal of this book is to give you a picture of what a successful modular web project looks like, and guide you through the pieces you'll need.

But first, let's take a step back. Why think about the web at all?

Why your website matters

The Internet is the most direct channel to your customers. Your website is the most universal way you can tell your story. It's the only channel you fully control.

Your website is **a source of information**. Like a commercial, it's a **brand artifact** giving you the chance to tell your story**.** It's **part of your customers' journey.**

For your demand generation team, your website is an important **customer acquisition channel.** It's your **storefront** (if you offer e-commerce); it's the **system of record** for content writers and editors and the **engineering infrastructure** for your developers.

As a result, websites are some of the most cross-functional projects within an organization. For some organizations, that means a long list of requirements tacked on by two dozen stakeholders, and a technology search process that will converge on the oldest, slowest CMS systems.

But design-by-committee is dangerous. The web moves quickly. What looked *great* two years ago looks *okay* today. (Let alone three, or five years.) The Internet is a crowded place, and it's tough to stand out.

A better idea: proactively searching for, experimenting with, and gradually rolling out, the *best* technologies for your particular organization.

There are a few challenges. The larger number of categories, and technologies within each category. Figuring out where to start. Overcoming inertia and skepticism. Building organizational capability. Figuring out which technologies work well together. Navigating exploding *combinatorial* possibilities.

These challenges, and questions, keep quite a few different types of people up at night. Executives overseeing web and digital strategy. Managers and agency leaders working on web and

digital projects. Technologists who want to bring their company, or clients, into the future.

If you feel like you're sifting through hundreds of puzzle pieces, without the picture on the box for reference – well, here you go.

You don't need any particular technical background to read this book. I've intentionally written for each of the many different roles who help build and operate websites – content creation, operations, and strategy; design; development; growth and demand gen; project management, and so on.

Nor do you need to be in any particular industry. I've written this book with examples and discussion of consumer goods companies, nonprofit organizations; brands and retailers, online publishers and organizations with complex content publishing needs, software companies; digital agencies, other B2C businesses, and so on.

What the modular web is

I wrote this book to introduce you to a new website architecture that's poised for growth in the 2020s. But a history of the web shows that the web's architecture changes almost every decade.

Back in the 1990s, websites were *handcrafted*. When you wanted to change your website, your webmaster would edit an HTML document and upload it to the server.

In the 2000s, custom *content management systems (CMSs)* became common. Agencies would build you a CMS, or add functionality on top of an existing CMS, tailored to your requirements.

In the 2010s, the CMS began to standardize. A core feature set formed: content modeling, tagging and categorization, editorial review workflows, page building, publishing.

CMS adoption tripled, reaching 60% of the web, with WordPress leading the way.

Now, in the 2020s, the *modular web* is emerging as a new architecture for websites. Every website role, trying to better serve customers, is adopting specialized cloud-based tools.

Headless CMSs. JavaScript frameworks. Jamstack frameworks. Build & deploy platforms. Design tools. Search. Auth. CDPs. Product analytics. Session recording. E-commerce platforms, with logistics, rewards, subscriptions, reviews….

This new architecture has a number of benefits – high-performance websites, fine-grained analytics, content re-use. It also requires certain skillsets to implement, and there are some workflows it doesn't excel at (yet). At the end of this chapter there is a diagram comparing the "performance envelopes" of the modular web with a traditional CMS. ①

This new architecture has many names: decoupled, headless, best-of-breed, Jamstack, composable, edge. I call it "modular", because it's about rethinking the building blocks we use to make websites:

What's in this book

This book covers a few core, repeated themes: the cloud, e-commerce and demand generation, content and the CMS, the mobile web and web performance, modern JavaScript and the Jamstack, web design.

Our team has had thousands of conversations with website practitioners exploring and implementing the modular web. With a tech lead or engineering manager, we might share website optimization tips or talk rollout strategy. With a VP Engineering or director of digital marketing, we found that landscape summaries, brief historical context, and architecture diagrams resonated more.

In our conversations, we found that content strategists and e-commerce gurus appreciated a primer on modern development; meanwhile, developers were more effective with a better understanding of marketing workflows and team goals.

We wrote this book as a shared reference for each audience – a business brief, tech landscape summary, and how-to manual.

Some topics we'll explore:

→ how mobile optimization has become essential to online success (Chapter 2)

→ how the CMS began to standardize – just as hundreds of cloud-based SaaS startups emerged to disrupt it (Chapters 3, 4)

→ how digital advertising and demand generation raised the bar for website performance (Chapter 5)

→ how JavaScript became the go-to language for web development (Chapter 6)

→ how mobile and JavaScript-based component libraries raised web design standards (Chapters 6, 10)

→ how the Jamstack became the standard for fast, scalable websites (Chapter 7)

→ how to build an effective modular web architecture (Chapters 4, 8)

→ retail and brand digitization in the COVID era (Chapter 9)

→ why different CMSs adopt headless at different rates (Chapter 10)

→ how mobile performance began to affect SEO (Chapter 10)

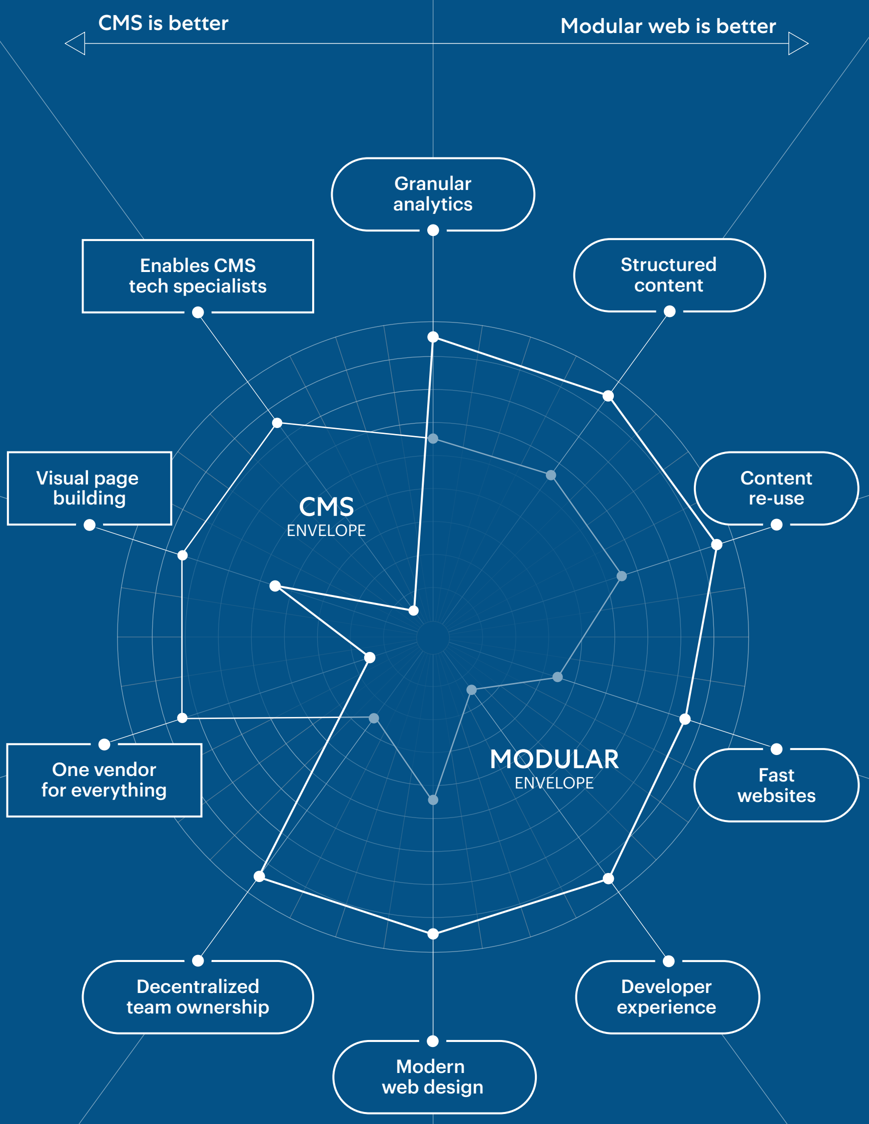
→ a map of all of the types of modular web technologies and how they integrate together (Chapters 8, 11)

→ how to empower modular web champions within your organization (Chapter 11)

→ the different journeys organizations take toward the modular web (Chapter 11)

→ how the modular web is likely to evolve (Chapter 12)

Welcome to the web's new architecture. I'm glad you're here.



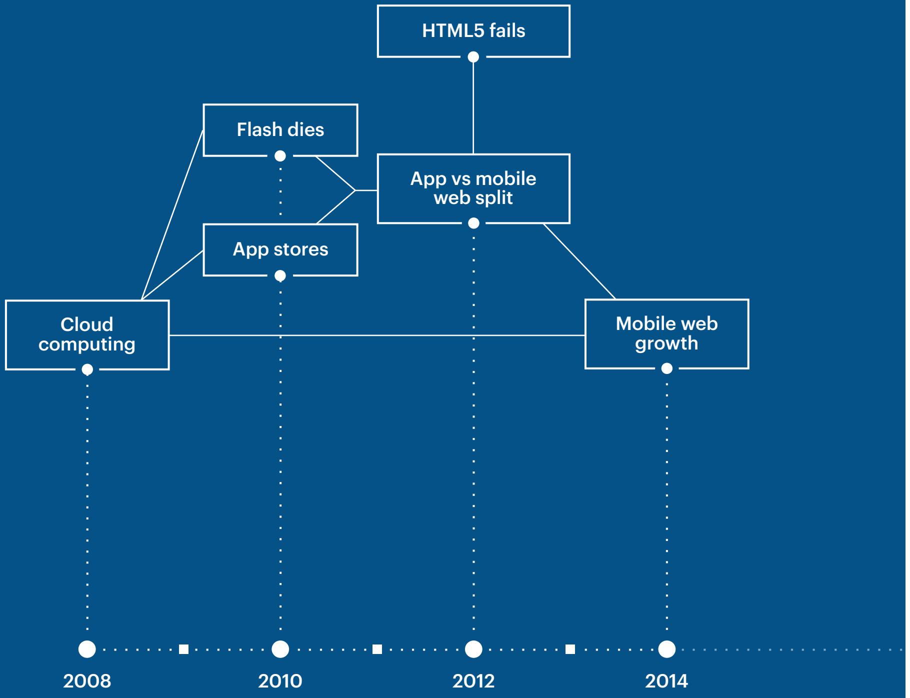
1 Modular Web vs Traditional CMS
Performance Envelope

MOBILE WEB

CHAPTER 2

MOBILE AND THE ATTENTION WAR

MOBILE CHANGES THE WEB • THE INTERNET INTENSIFIES COMPETITION



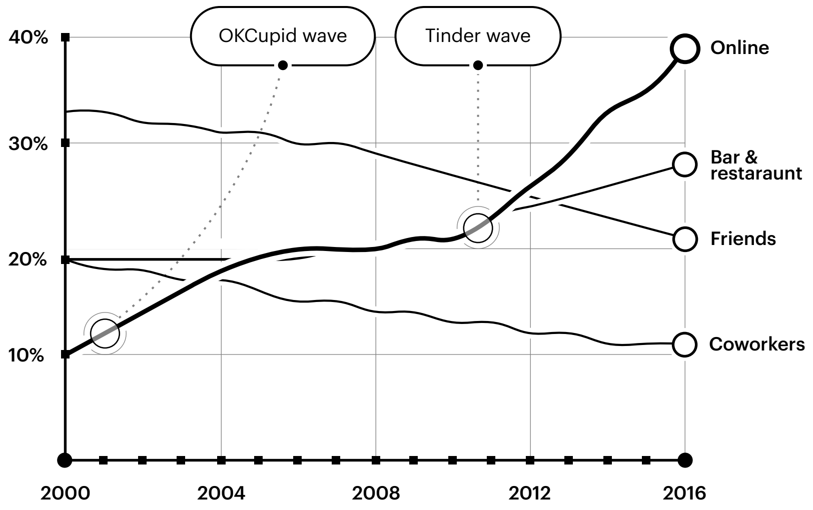
Consider one of the most important choices we make in life: our choice of partner.

A Stanford research study surveyed 5,500 Americans and asked them *how* and *when* they met their partner.

The four most common methods for couples who've met since 2000: online, in a bar or restaurant, through friends, and through work.

During the *Friends* era, the top way to meet your partner was, in fact, through friends. "Online" stayed mostly flat during the mid-to-late 2000s – but surged ahead around 2010. And by 2017 almost 2 in 5 couples were meeting online.

Why? Smartphones.



❷ How US heterosexual couples met

The word "mobile" gets applied to smartphones – but maybe that's a misnomer. After all, it's *human beings* that are mobile. Smartphones are just computers that accompany us as we move.

"If you look at *every other technology* ever invented," then-Facebook engineer (and later Reddit CEO) Yishan Wong wrote in 2010, "the ability to use it in arbitrary locations vastly increases its usability."

The "OKCupid wave" and the "Tinder Wave" give us a snapshot of the desktop and mobile eras of the web.

And in business – as in romance – an internet that goes where we go opens up new worlds of connection, innovation, and of course, competition.

Mobile begins to change the web

As smartphones became popular, they quickly began to change the web. Adobe Flash, which powered most web animation and interactivity, drained battery life and missed the mobile transition. A set of browser APIs known as HTML5 began to go live, including features like "drag & drop" file upload, browser data storage, and animated graphics.

But after some large bets placed on HTML5, most prominently by Facebook, failed, it became clear that the web's capabilities would take time to fully mature.

Steve Jobs launched the iPhone in 2007, but the App Store didn't launch until 2008, following the emergence of an underground "jailbreak" community. By the early 2010s, people's behavior had started to solidify; they were using native apps in different ways than the mobile web.

Native apps were faster, with fewer taps. So people used apps for camera/GPS *(maps, video messaging)*, messaging

friends *(email, messaging, dating)*, and immersive experiences *(mobile gaming, movie streaming, watching sports)*.

But when you needed breadth rather than depth, you tended to stay on the mobile web: shopping, searching for information, handling business, finance, real estate, or auto-related activity.

The early mobile web suffered from constraints: small processors, slow connections, sites with layouts not yet adapted for mobile. Users abandoned pages on mobile far more than on desktop.

The war for attention

As smartphones took off, people began spending more and more time on their devices, and mobile traffic grew with it.

Mobile devices rose from near-zero to 20% of web traffic by 2013. By 2016, they had reached 50%.[3][4] Increasingly, consumers were surfing and buying from their smartphones – while riding public transportation, waiting in line, escaping from awkward situations.

As mobile made the Internet ubiquitous, the war for attention heightened. In the analog world, most of your options were within driving distance. On the Internet, you could be anywhere, instantaneously. And with larger markets came heightened competition.

The Internet's attention war had winners and losers. The winners: mobile gaming; social media; streaming companies; Amazon. The losers: Borders, Radio Shack, Circuit City, ToysRUs, your local newspaper, your local mall.

The Internet was a melee. The web competed with native apps. Content and news competed with Facebook and Twitter. Brands, stores, and retailers all competed with Amazon, and with each other.

Organizations competed for SEO juice, viral posts, ad inventories on target demographics, placement on search terms, retargeting, influencer endorsements, PR, space in your inbox....

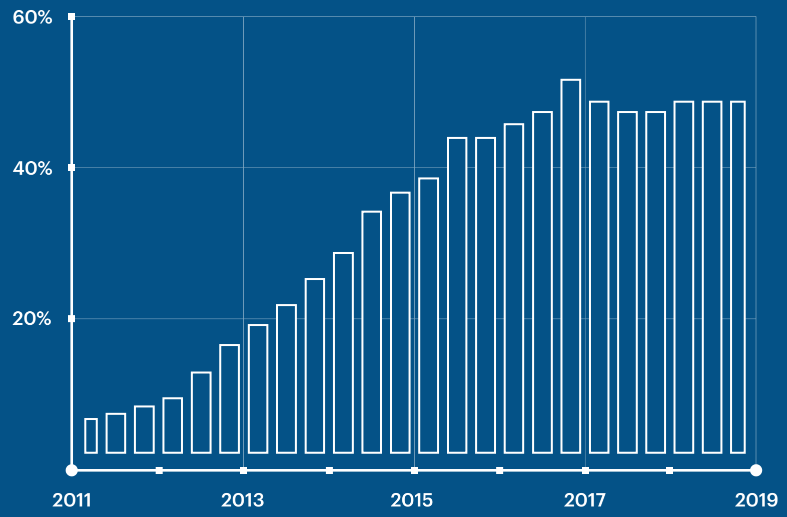
The more an organization depended on the Internet, the more acutely they were aware of the new reality.

Key takeaways

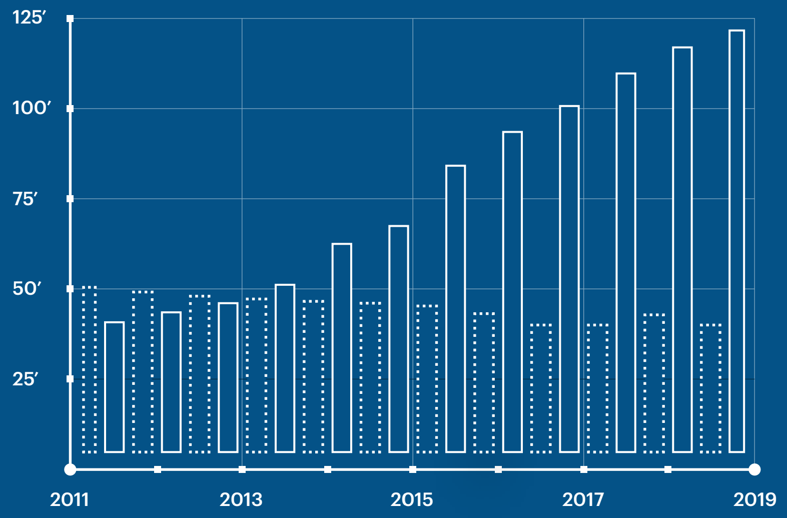
The smartphone fundamentally changed how we use the Internet.

Today's Internet is incredibly crowded, and it's difficult to stand out.

Companies that don't optimize mobile web performance risk losing up to half of their traffic.



3 % of web traffic from mobile



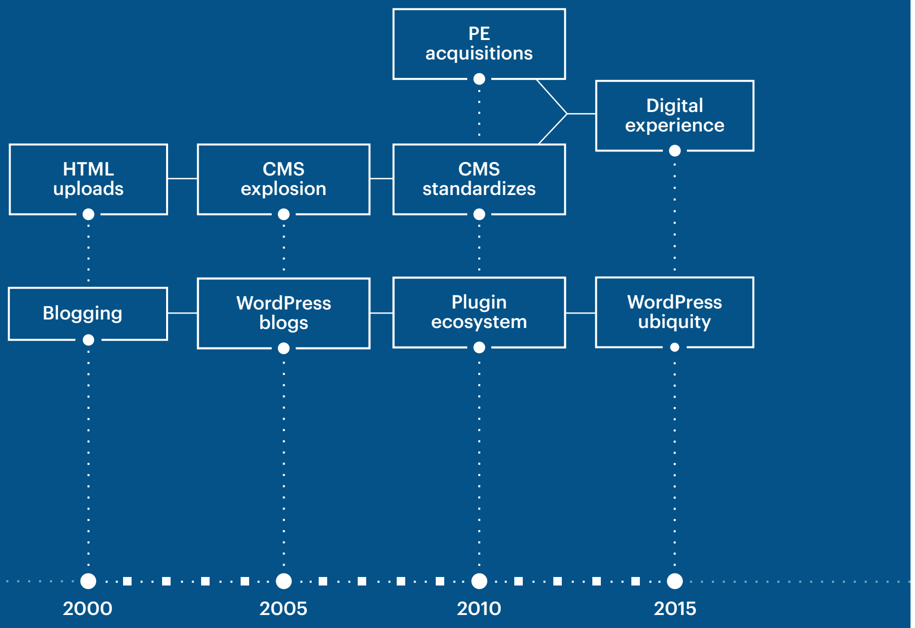
4 Daily time spent with the internet
in minutes ⁞ desktop ☐ mobile

CMS

CHAPTER 3

THE CMS STARTS TO STANDARDIZE

THE EVOLUTION OF ENTERPRISE CMSS • THE RISE OF WORDPRESS • THE CMS ADDS MARKETING TOOLS



In the early 2010s, most organizations were using a content management system (CMS) to build their web presence.

CMSs come in a lot of different flavors. But at their core, CMSs do four things:

→ **Modeling:** Structure and store content
→ **Aggregation:** Tag and categorize content
→ **Editorial:** Allow users to create, edit, and delete content
→ **Publishing:** Get content on the web.

The early CMS as a customizable system

In the early days, websites were handcrafted HTML pages, manually uploaded to a server. But when media organizations and corporations started using the web, they needed systems to handle content in bulk (dozens of news articles per day!), or in complicated structures, hierarchies and taxonomies.

This gave rise to the content management system. In the beginning, companies handled this the way they'd handle another business process they needed software for. They'd head over to IT, or ask an external agency for a custom application.

By the early 2000s, a wide variety of CMSs had emerged. Many came from web agencies. Some were built in Java; others in .NET, C# or PHP. Some targeted specific sectors, like health-care, universities, or media organizations; others tried to build industry-agnostic tools.

By today's standards, these early CMSs offered a patch-work of functionality, and often relied on partner implementation shops to add key features – analytics, forms, search.

Veteran practitioners recall those days with rue and fond-ness. "Vignette kinda sucked," laughs content strategist Larry

Swanson, referencing the dominant CMS of the late 90s.

WordPress moves from blogging to everything else

Around the same time, the open-source LAMP stack emerged to fuel the growing blogging trend – hosted sites like Blogger (1999) and LiveJournal (2000); customizable open-source CMSs like Drupal (2000) and WordPress (2003).

For the ordinary Internet user, blogging was their first (and only) experience with a CMS. WordPress built an intuitive way to write content, and beat out other blogging systems because its themes allowed users to personalize their web presence.

As it grew, this ecosystem of themes and plugins helped turn WordPress from a personal blogging platform into a tool for business websites and online publishers.

WordPress developed a wide selection of *business-specific themes,* for auto dealerships, realtors, florists, and so on. Plugins provided CMS features like forms, analytics, SEO, marketing automation. S*ite building tools* allowed developers to build interfaces via drag and drop UIs.

Almost any kind of small or medium business (SMB) could build a web presence relatively quickly with the help of a local WordPress developer.

By 2011, when the first web technology statistics were collected, standardized content management systems were just under a quarter of the web – and over half of that was WordPress. ⑤

The CMS bolts on marketing tools

Though WordPress took over individual blogs and SMBs, it

didn't really make much of a dent with larger organizations. By then, the "enterprise" feature set had solidified, including things like structured content and digital asset management.

Larger companies needed these specialized features – and they weren't well-supported in WordPress. Sure, a few digital publishers – think *Huffington Post* – still gravitated to WordPress. But larger companies tended to stick with a familiar enterprise CMS.

With content use-case boxes checked, the enterprise CMS started to move into marketing tools. In the mobile era, companies wanted to identify customers across devices ("omnichannel"). They wanted to integrate their content with commerce, and to integrate both of these with ad targeting and analytics systems. Larger CMS vendors started purchasing adjacent systems – for personalization and targeting, analytics and attribution, managing digital assets, doing light integrations, and selling them to customers alongside the CMS.

Adobe's CMS, for example, had been built by a Swiss company, Day CQ, that Adobe purchased in 2010. Adobe rebranded the software as "Adobe Experience Manager", selling it alongside "Adobe Analytics", a re-branding of Omniture, a Utah-based company they'd purchased two years earlier.

As the mobile era progressed, CMS vendors consolidated – from lots of small software shops into a few large public companies (Adobe, Oracle, HP) and larger CMS companies (Sitecore, Episerver) owned by private equity firms.

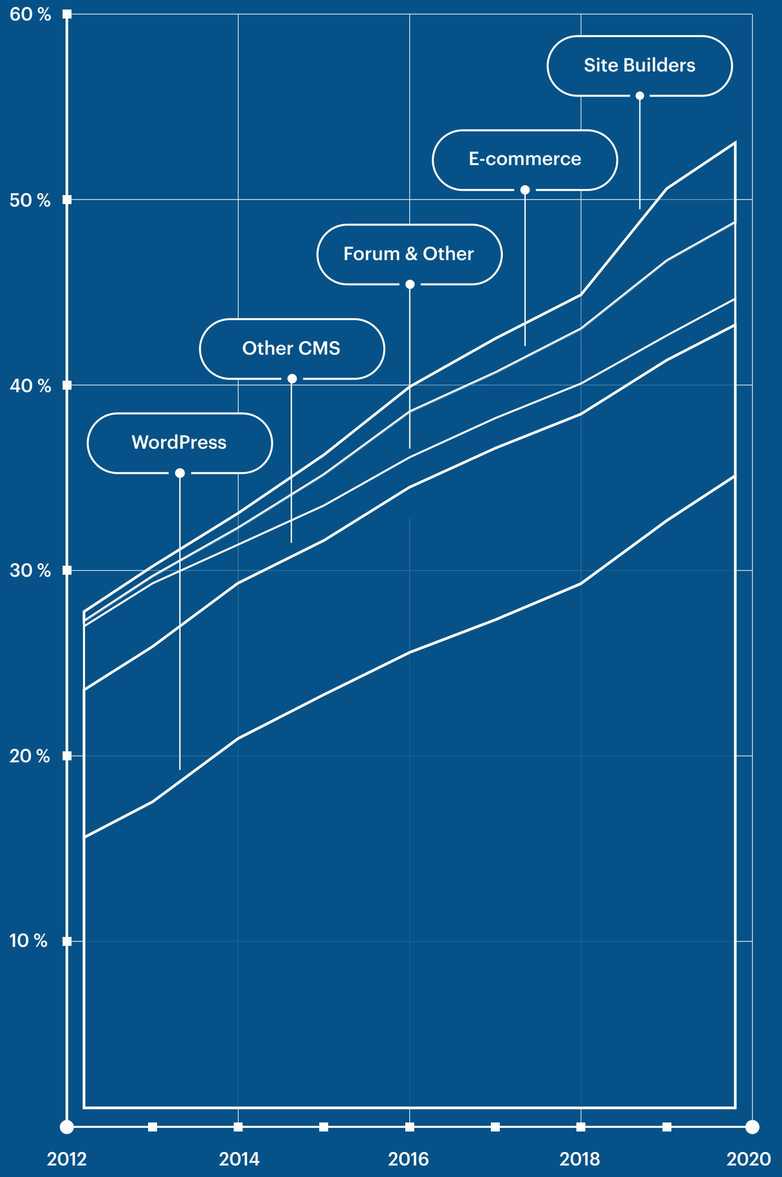
Key takeaways

At their core, CMSs allow users to create, organize and publish content.

Today's CMSs were born in the pre-cloud era, and still carry over the architecture of that time.

Over time, features like personalization, analytics, and form building have grown somewhere between CMSs and marketing systems they integrate with.

The CMS has become not just a website tool but a marketing tool.

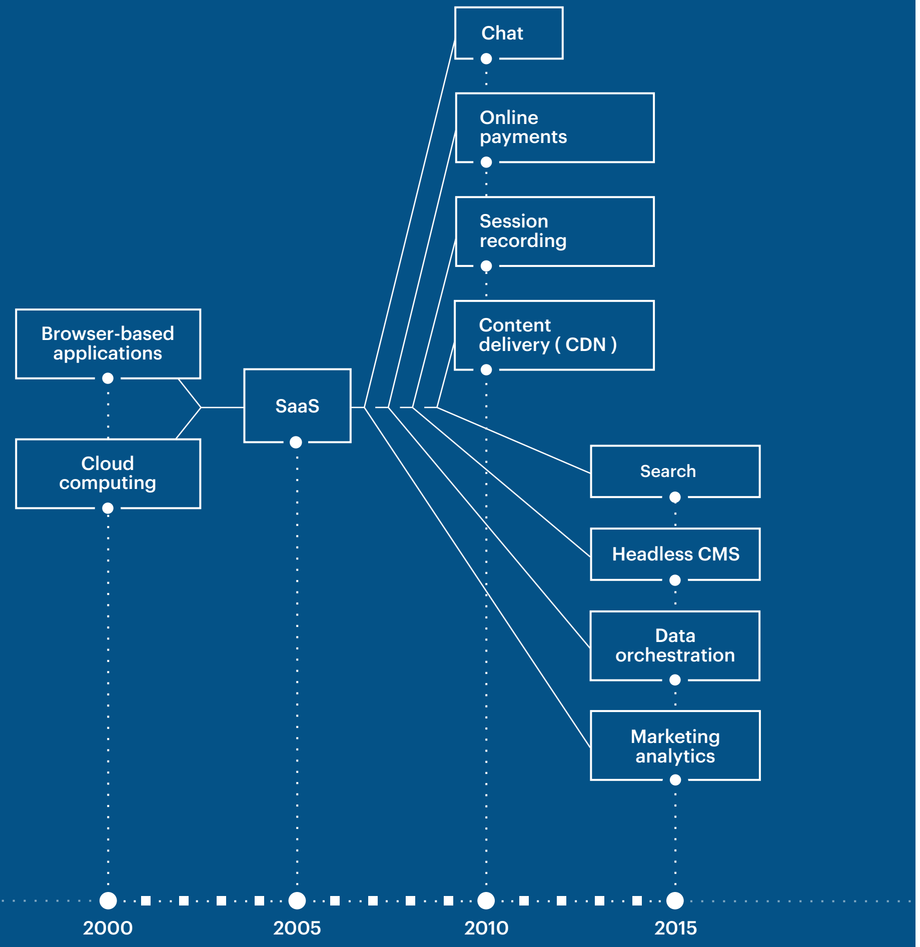


5 **Web Technology Growth**
W3Techs top 10 million websites

CHAPTER 4

CLOUD COMPUTING MEETS THE WEB

SILICON VALLEY'S SAAS EXPLOSION • STRIPE, CONTENTFUL, AND SSGS



As the internet got bigger, it started to break a key architectural assumption of the early web: that a single server could handle all traffic and user requests originating from anywhere in the world.

The birth of cloud computing

Scaling always-on software services has, historically, been difficult. Complexity abounds – from the physical challenges of maintaining server racks, to the software challenges of networking, load-balancing, failovers, backups.

In the mid-2000s, a wave of services promised to leave these problems behind.

Amazon Web Services, Google Cloud Platform, and Microsoft Azure – along with smaller players like Heroku or DigitalOcean – promised developers a world where they could stop worrying about infrastructure and systems administration.

From the perspective of engineering or IT, these services simplified planning. No more trying to project capacity and budget for servers months in advance. Spin up servers when you need them; turn them off when you don't.

The cloud changed applications architecture. Instead of running in a data center behind a firewall, applications were run in the cloud, accessed via web browser on the public Internet.

TECH FEATURE

[Payments] Stripe

The "aha" moment came during a wistful conversation.

Two teenage Irish hackers, Patrick and John Collison, were reflecting on the projects they had worked on over the years. The brothers were unusual in several ways: their youth, their talent, their commercial sense. The brothers had grown up in Ireland; the previous year (2007), within the course of 10 months, they had built and sold a company for $5 million.

Reflecting on half a dozen side projects – jailbroken iPhone, a web framework – the Collisons realized that even though some of their projects had become quite popular, they'd never been able to collect payments. They'd tried to create merchant accounts at PayPal, but the application process dragged on for weeks, and they gave up.

The Collisons were attending school in Boston – Patrick at MIT, and John at Harvard. Faced with a cold winter break, the brothers flew to Buenos Aires, where the weather was warm and they could work at cafes through the night. Over a month, they hacked together a prototype for a system allowing businesses to easily add payments, modeling legal and regulatory requirements like merchant gateways and PCI compliance laws.

The Collisons returned to campus for spring, but in summer they decided to focus on their project full-time. Hiring a few friends, they built a product, and in fall 2011, launched Stripe to the public.

Gmail and Dropbox pioneer browser-based apps

One early browser-based cloud application was Gmail, which launched in 2004. Part of Gmail's draw was better storage and spam detection. But the Gmail application was also much faster than Yahoo! or Hotmail.

When a user opened an email from their inbox, Gmail used new JavaScript features to just fetch new email's content – rather than reloading the whole page. Power users liked speeding through their inbox.

A few years later, an app called Dropbox began to take off. Dropbox used the cloud to sync files between devices. For files worked on by more than one person, this prevented endless hassles: editing the wrong version, losing changes, conflicting versions, filenames like 0419_presentation_3_LAST_FINAL_FINAL.ppt.

Silicon Valley creates thousands of SaaS services

Like Gmail, Dropbox represented a different model of software delivery and pricing.

Traditional business software was sold at a fixed price in the tens of thousands of dollars for annual or perpetual licenses; Dropbox priced per-user, for $9 per month, with a free-trial period to entice new users ("freemium").

Traditional software was installed in corporate data centers; Dropbox was hosted in the cloud. Traditional software required IT approval and maintenance; Dropbox's payment structure was explicitly designed to bypass IT departments. Traditional software required a sales team; Dropbox was designed to spread virally within companies.

Dropbox didn't invent cloud-based delivery or freemium

Contentful

After he shut down one startup, Berlin-based programmer Sascha Konietzke decided to freelance as a mobile developer, building iOS and Android apps. After a few projects, Konietze noticed that he seemed to run into the same problem again and again -- it was really difficult to manage the content that powered these apps.

At first, Konietze tried using CMSs suited for structured content, like Drupal, but found that it was very difficult to make these systems output the raw underlying content instead of HTML pages. He talked to other freelancers, but nobody seemed to have a good solution to this problem.

Some would store content in spreadsheets, but these were difficult to version and didn't have the right editing interface. Others quickly hacked together ad-hoc web services. These usually worked well enough to ship the app, but could go down months later for any number of reasons – traffic spikes, problems with shared hosting, and so on – prompting panicked calls from clients.

What these freelancers really wanted, Konietze concluded, was a web service to store content with a simple interface. It would need to be usable by clients, the freelancers wouldn't have to maintain it, and it wouldn't go down. Konietze signed up a co-founder, Paolo Negri, to create a cloud-based CMS specifically for mobile developers.

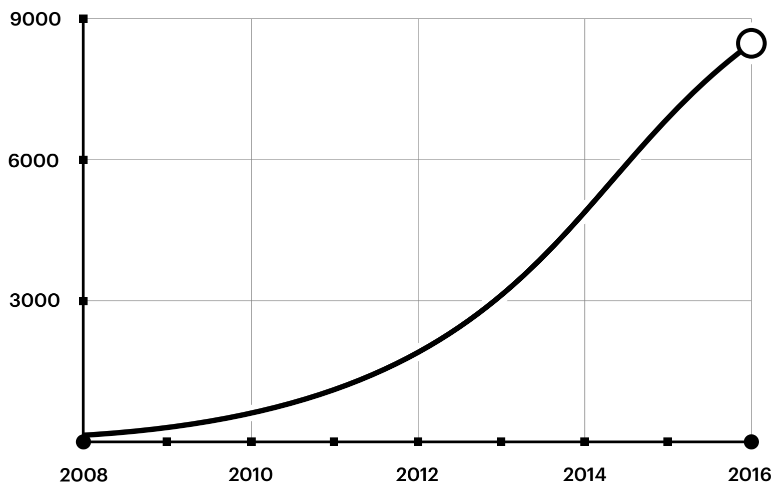
In the weeks after launch, hundreds, then thousands of freelancers and digital agencies signed up, and in 2013, Konietze and Negri re-architected and re-launched their CMS as Contentful, an API-first CMS.

SaaS pricing. But in 2010, when it approached hundreds of millions of users and a $4 billion valuation, the Silicon Valley hivemind kicked into gear.

Experienced VC moneymen swapped notes and coined buzzwords. Founders swapped ideas in Palo Alto cafes and talked about "disrupting" industries in Sand Hill Road pitch meetings.

Thousands of startups shifting billions of dollars of spending from on-premise software to the cloud. Business workflows (Docusign, Zoom). Financial technology (Square, Expensify). Project management (Airtable, Asana). Developer tools (Github, Datadog). Marketing technology (Hubspot, Marketo).

By 2015, the number of SaaS companies receiving venture funding had grown 10x, most building applications that ran in the browser. ⑥



❻ **Venture capital-backed SaaS companies**
Cumulative total

TECH FEATURE

Static site generators

In 2008, when building his blog, Github co-founder Tom Preston-Werner created a tool called Jekyll. Jekyll took text written in a format called Markdown, and outputted HTML, CSS, and JavaScript files.

Preston-Werner called Jekyll a “static site generator” and lots of developers moved their blogs over: AWS CTO Werner Voegels, programming Q&A forum StackOverflow. Static sites appealed to developer aesthetics around simplicity. There was no need to keep a database up, optimize web servers, or worry about a runtime.

Static sites had an alternative architecture to CMSs. When a CMS serves a website page to a visitor, it runs application code, and looks up information in a database before it returns a response. This allows customization, but adds delay. SSGs compiled content to “static” HTML files that were the same for every visitor. As a result, they were fast to load.

The static site world was reminiscent of an earlier hacker culture, captured in shows like AMC’s Halt and Catch Fire but which had faded from much of the developer world. It attracted engineers with high conscientiousness and a particular kind of taste, willing to modify or create new tools if the existing tools didn’t feel quite right.

Webtech: plug and play SaaS apps

One SaaS subsector was webtech: single-purpose applications that could be plugged into websites.

Webtech included API-based content management (Contentful), online payments (Stripe). authentication (Auth0), A/B testing (Optimizely), fine-grained analytics (Heap), session recording (FullStory), and search (Algolia).

A previous generation of Silicon Valley web startups – Google, Facebook, Twitter – had built for consumers. But these companies were building for developers, and for businesses.

Architecture: CMS vs Cloud

In the early 2010s, webtech had far fewer features than the enterprise CMS. But three advantages would let it catch up quickly:

First, higher shipping velocity. By the early 2010s, most cloud-based app development teams had embraced practices like version control and continuous integration and deployment (CI/CD). Most CMS development teams would take a decade (or more) to adapt.

Second, faster feedback cycles. Cloud “multi-tenancy” meant that updates could roll out instantaneously across all users, allowing teams to quickly start working on the next thing. Meanwhile, on-premise CMSs had to wait for risk-averse customer teams to upgrade to new versions.

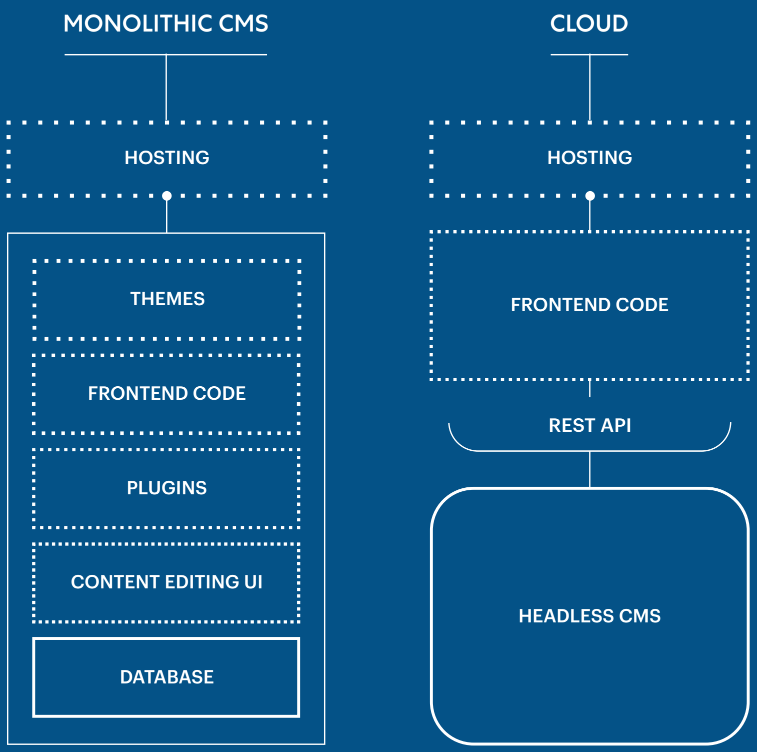
Third, massively parallel development. In this arena, one hundred duck-sized horses beat out one horse-sized duck. Each category – search, content management, analytics, and so on – could evolve on its own, with dozens of startup teams working independently. “Divide and conquer” meant moving faster.

Key takeaways

In the 2010s, cloud computing changed how software is built and software-as-a-service (SaaS) changed how businesses consumed it.

Cloud-based plug-and-play SaaS apps for websites emerged: webtech for content management, payments, web analytics, search, etc.

Cloud architectures enabled faster development and more rapid iteration than pre-cloud architectures (like the CMS).



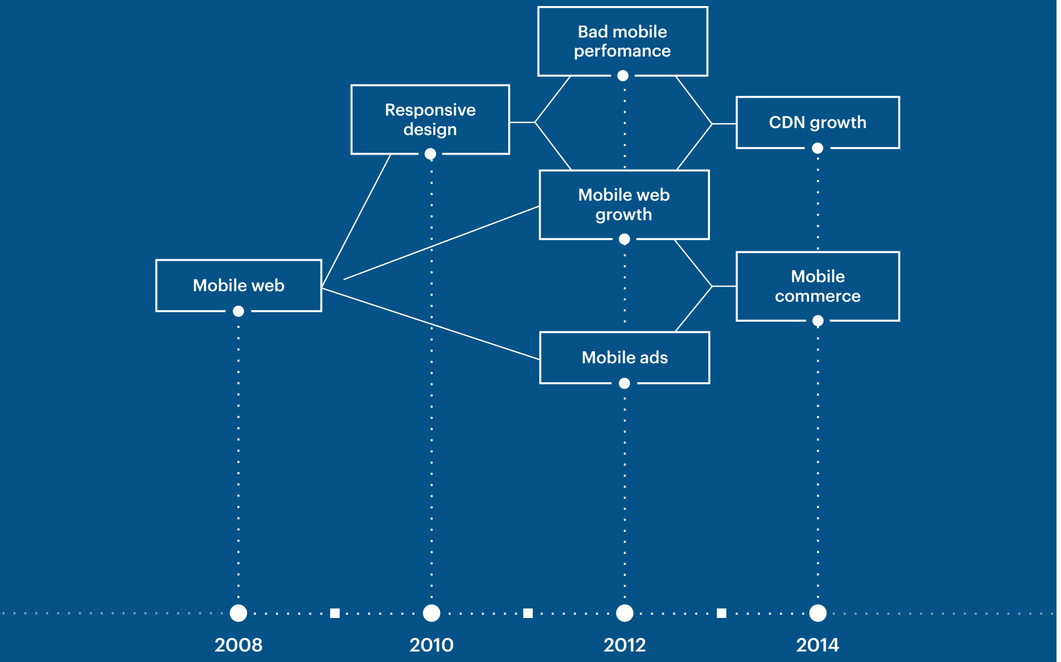
HOMEGROWN / POORLY INTEGRATED VERSION CONTROL →	CLOUD-VERSION CONTROL (GITHUB), AUTO-LINTING, FORMATTING
MONOLITHIC, RISKY RELEASES →	CONTINUOUS RELEASE VIA CI/CD, DEVOPS TOOLING FEATURES FLAGS MULTIPLE VERSIONS OF COMPONENTS
ON-PREM INSTALLATION, MANUAL UPGRADES REQURED BY IT →	CLOUD-BASED MULTI-TENANCY, AUTOMATIC UPGRADES PERFORMED BY VENDOR

7 CMS vs Cloud architectures

MOBILE WEB

CHAPTER 5 THE MOBILE WEB BECOMES A FUNNEL

MOBILE ADS TAKE OFF • MOBILE COMMERCE AND PERFOMANCE • CDNS

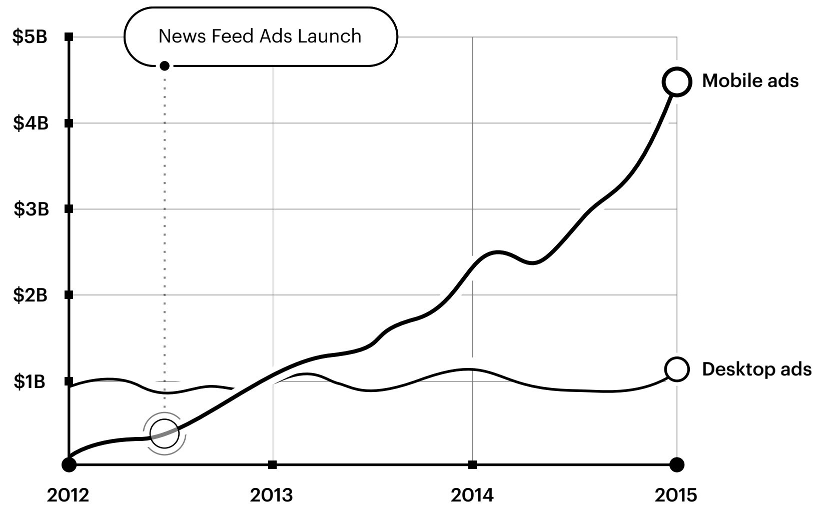


One of the first groups to notice the mobile web was advertisers.

Social ads become mobile ads

In late 2012, Facebook was in trouble. Following its IPO, its stock had dropped by 50% in only a few months. The main problem from Wall Street's perspective: what good is a billion users if you can't make money?

"What saved Facebook's stock [price]," wrote former Facebook ads product manager Antonio Garcia-Martinez, "was ads in News Feed, while the user was on his or her mobile device. That's it." (8)



(8) FB quarterly revenue by device type

To work well, successful ads products (say, sponsored Google search results) need good display formats, and good targeting. To get users' attention, an ad needs to get in a user's visual field *and* be relevant.

Social media companies had access users' demographic information, and, when users logged in from multiple devices, could cross-link browsing data. This turned out to be enough to create relevance: clickthrough rates on mobile Facebook News Feed ads were incredibly high.

Facebook's mobile ad revenue grew by 20x in three years, quickly surpassing desktop ads. Google quickly followed suit, with mobile ads surpassing desktop ads by 2015. The creator of News Feed ads, Facebook product manager Fidji Simo, would go on to run the entire Facebook app before becoming the CEO of grocery delivery service Instacart.

Mobile commerce spawns DTC brands

While they raised significant privacy concerns, mobile ads were a huge boon for digital commerce. They helped existing brands to *reach* their audience – and upstarts to *create* one.

In 2007, Andy Dunn was fresh out of business school when he started Bonobos, an online men's jeans company. Dunn credited Facebook ads for early sales growth and continued to use them heavily as smartphones took off, driving over 10% of traffic from Facebook ads.

Curved waistbands and a hip millennial aesthetic got Bonobos into Macy's and Nordstroms. The company expanded to shirts and suits. By 2016, Bonobos employed over 300 people; a year later, Walmart bought the company for $310 million.

Along the way, Dunn started a conversation about a new kind of brand – born on the Internet, "aimed squarely at millennials and digital natives" – that became known as "direct-to-consumer," or DTC.

DTC brands were a departure from the half-century-old discipline of brand creation. The brands we know as consumers – in food, beverages and beauty, and longer-lasting products like clothing, home furniture, and appliances – were mostly created around mid-century by a few large consumer companies.

These corporations – P&G, Unilever, and Nestlé, used a centralized process including logos, pricing, packaging, positioning. Then, as *Mad Men* watchers will know, they leaned on the Don Drapers of the world for mass media ads, and their retail relationships for distribution clout.

Few other brands took off. The bottleneck they faced: shelf space. Winning over retailers could take years if not decades.

But with mobile ads, DTC companies could end-run around retailers and reach consumers directly.

"The ease of opening a business on Facebook has in turn spawned a wild proliferation of specialty digital sellers," wrote Burt Helm in the New York Times. "Many of them follow the same playbook and even share a similar aesthetic."

From 2010 to 2015, mobile commerce grew over 10x – from $3B to almost $40B in the US.

But by 2015, even though visitors split equally between mobile and desktop, desktop still represented 85% of online purchase dollars.

The ongoing mobile performance bottleneck

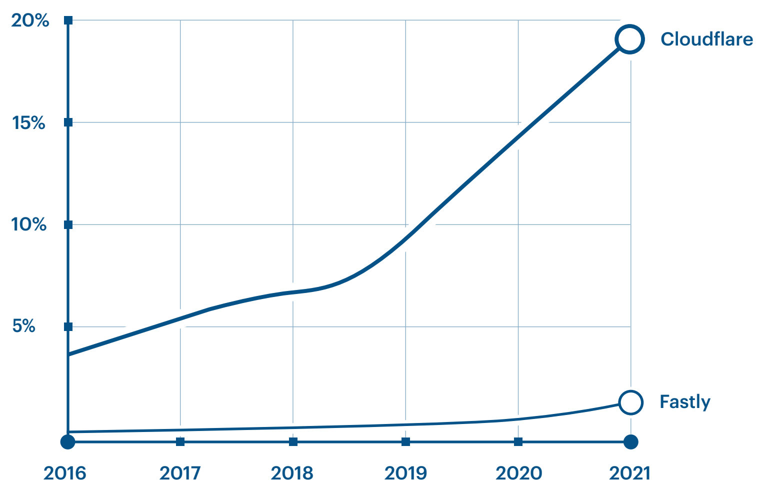
Mobile was working for product *discovery*. But to be optimized for e-commerce, the mobile web needed to make *evaluation*, *comparison*, and *purchase* easier. By 2015, mobile sites were mostly *responsive*. Making them *fast* was trickier.

The CDN: Cloudflare, Fastly

Before the cloud, enterprises served website traffic from servers from their data centers. To secure the network from malicious traffic, they bought and configured expensive, customized network hardware: VPNs, firewalls, routers, load balancers.

After the cloud, these sorts of services became much more widely available and far cheaper. Services like Cloudflare and Fastly (9) launched with a content delivery network (CDN) that would cache content at multiple "points of presence" around the world. They also included protection from malicious distributed-denial-of-service (DDoS) attacks.

With content closer to end-users, latency dropped. An Australian visiting an American website could be served a local copy, without their request having to make a roundtrip across the Pacific.



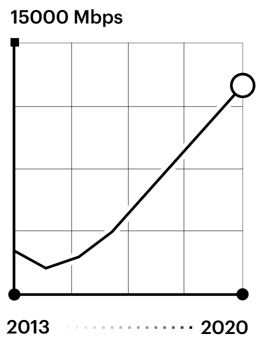
9 CDN usage as % of web

As phones and networks got faster, websites got bigger. One culprit was third-party code, which became far easier in the 2010s. JavaScript developers installed new packages via npm. WordPress developers loaded up their sites with plugins. Marketers used Google Tag Manager to drop in new scripts from martech vendors.

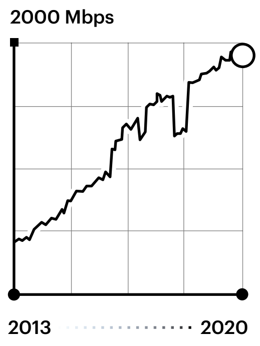
But performance was like a hamster wheel – running quickly, going nowhere. (10) (11) (12)

Website owners turned to the cloud, rolling out content delivery networks (CDNs) to reduce load times. Hardware and networks improved: phone manufacturers shipped faster processors; telcos rolled out 3G and 4G.

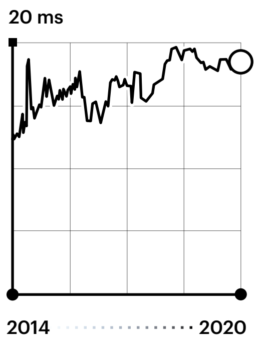
Another problem: workplace desktop wi-fi or Ethernet Internet connections made slow mobile page loads largely invisible to website teams.



10 Average mobile con. speed



11 Median mobile page weight



12 Average mobile load time (3G)

The new mobile marketing paradigm

The mobile paradigm shifted the character of marketing departments.

With the increase in web eyeballs and surge in digital advertising, marketing spend moved away from brand advertising on Sunday afternoon football games, and toward trackable, quantifiable digital ads and marketing campaigns.

Almost every dollar spent on mobile ads sent visitors clicking into the mobile web, with a demand-gen marketer on the other end measuring how well the ad channel was converting into customers.

By 2016, almost a third of all ad dollars were being spent digitally, with two-thirds for every other channel combined – TV, radio, print, billboards, and so on. The smartphone's timesucks – social media, mobile gaming – were all being monetized by ads pointing to the mobile web. (13)

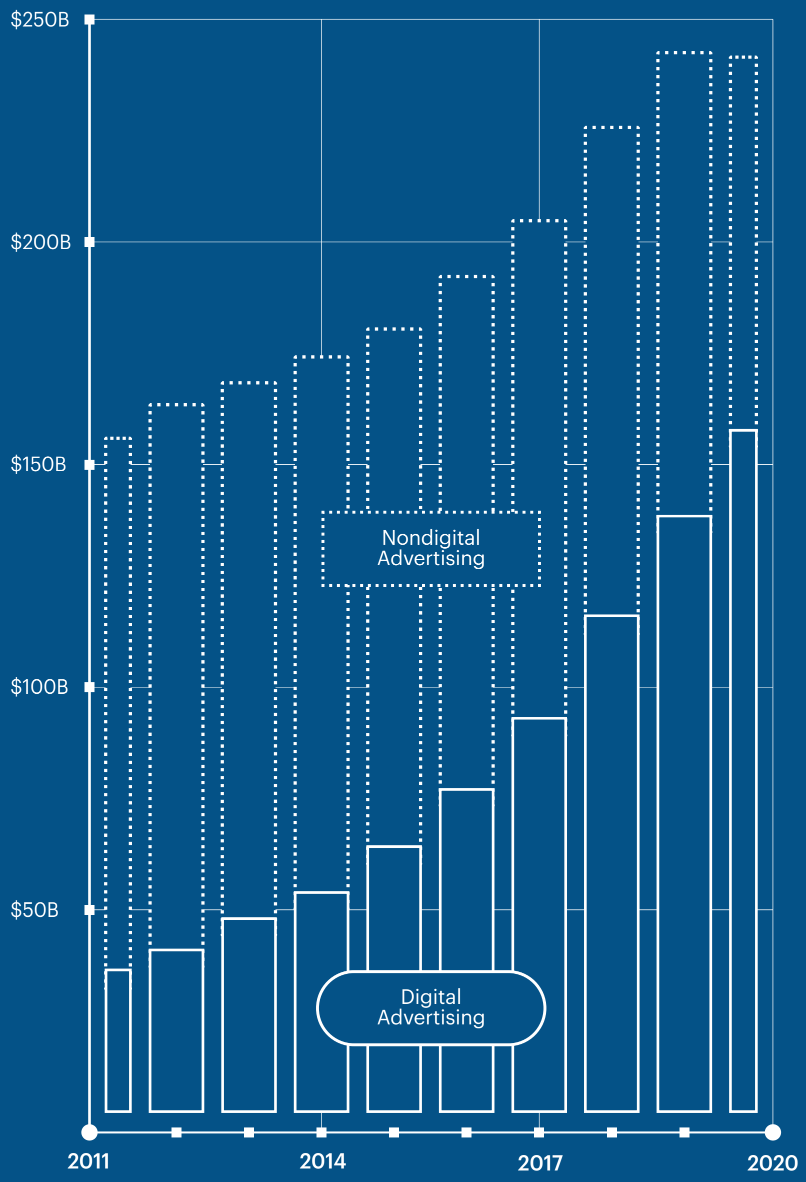
Key takeaways

During the 2010s, paid advertising became mostly about digital ads, and digital ads became mostly about mobile ads.

New, online, upstart brands used mobile to capture market share from established brands and channels.

Optimizing mobile funnels became the key growth challenge for online businesses.

Slow site speed became the key bottleneck in optimizing mobile conversion.

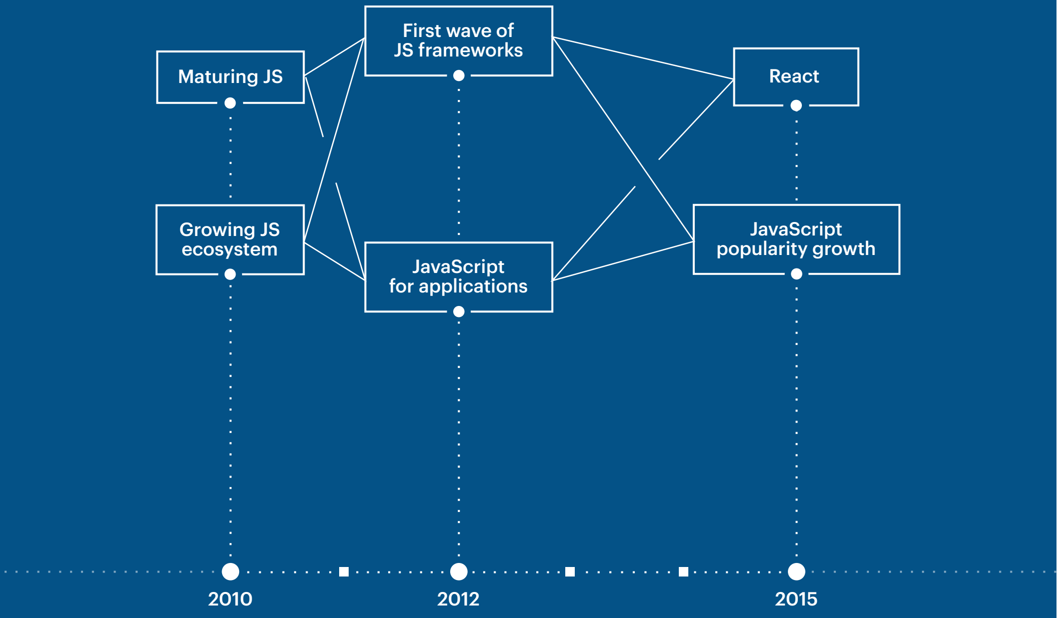


13 US advertising spend by channel

CHAPTER 6

JAVASCRIPT BECOMES THE UNIVERSAL LANGUAGE

ALWAYS BET ON JAVASCRIPT •
THE RISE OF REACT • COMPONENT LIBRARIES


Maturing JS
First wave of JS frameworks
React
Growing JS ecosystem
JavaScript for applications
JavaScript popularity growth
2010
2012
2015

JavaScript is the *lingua franca* of the programming world. Like English, it's a polyglot language, full of loan words from other cultures. It's not perfect, but it's everywhere. JavaScript started in the browser, but now it's used on the server (Node.js), in desktop apps (Electron) and mobile apps (React Native), and in place of CSS and HTML (React's JSX). The most common format for data transfer, JSON, comes from JavaScript.

The JavaScript ecosystem is massive – it has more third-party libraries than any other language, by a factor of 1,000. And it often seems disorganized and sprawling.

Frameworks and the early JavaScript ecosystem

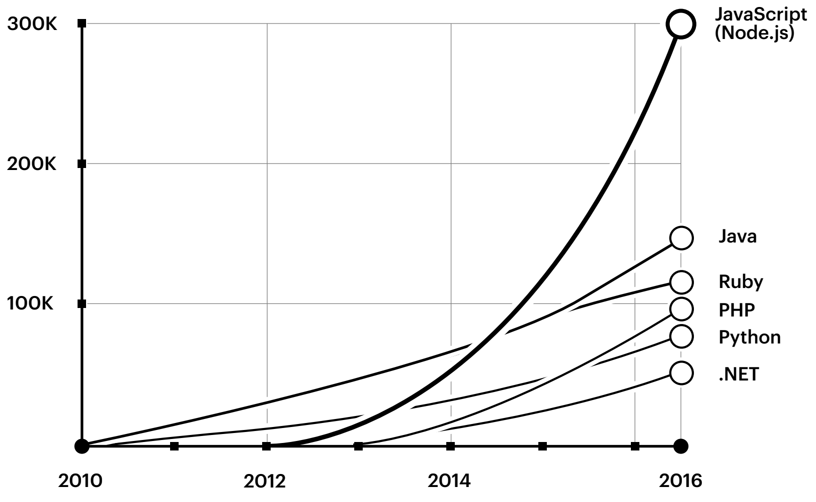
But let's rewind the clock. In the late 2000s, JavaScript was not a great programming language. Its odd syntax, cross-browser issues, and exclusive web use earned frequent mockery. Its origin story didn't help – JavaScript had been created in 1995 by Netscape programmer Brendan Eich in a ten-day, sleep-deprived, caffeine-fueled frenzy, trying to ship it before Microsoft launched Internet Explorer.

But JavaScript was changing quickly. "The second age of JavaScript started with the *annus mirabilis* of 2009," says prominent developer advocate Shawn Wang, pointing to the birth of Node.js (server runtime), npm (package management), and ES5 (better syntax) in that year. In a 2011 talk, Eich personally rebutted JavaScript's detractors at a meetup talk, highlighting recent progress.

While JavaScript was growing fast, its promise could outrun its capabilities – especially for application developers. JavaScript code interacting with multiple, inter-dependent third party services often led to *"callback hell."* Deeply nested UIs created *"jQuery spaghetti"* with tangled data manipulation code.

In the early 2010s, developers turned to open-source frameworks like Meteor, Ember, Backbone and Angular. These frameworks – and there were dozens – were often created by individual developers and launched on the dev forum Hacker News. A StackOverflow analysis concluded that a new framework would become "hot" every six months, but typically began fading twelve to eighteen months later (with lessons and patterns from earlier frameworks surfacing in successors).

As its ecosystem grew, JavaScript gained features of more mature languages: richer syntax (ES6, ES7), "bundling" tools (webpack, babel), "linters" for code style (eslint, prettier), typing systems (Typescript), testing frameworks (Jest, Cypress). Soon, it became common for a small team to build a whole application frontend in JavaScript.

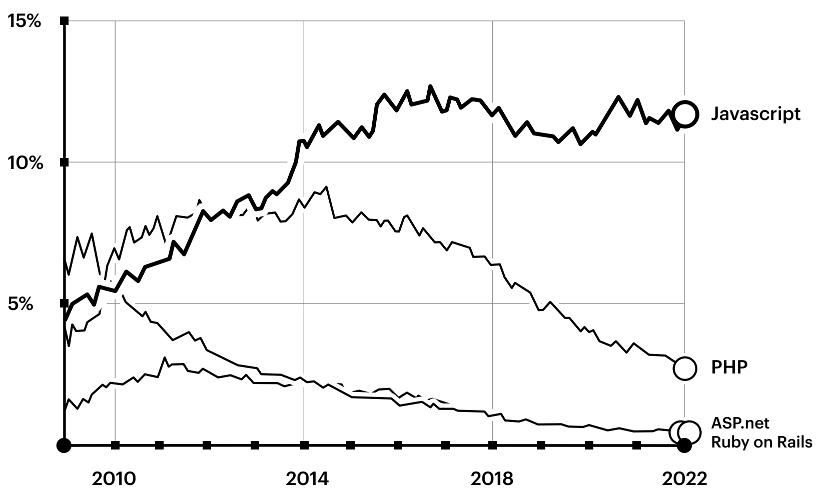


14 **Ecosystem size (in modules) by language**

"Always Bet on JavaScript"

Just as most people carry the musical tastes of their teenage years well into adulthood, most developers prefer technologies that were popular when they entered the industry.

Between 2005 and 2012, the most popular web development languages were PHP, Ruby on Rails, or ASP.NET. But by 2015, JavaScript had taken over. (14) (15) More and more, web developers became JavaScript developers.



15 **Web development language popularity**
% of StackOverflow questions tagged

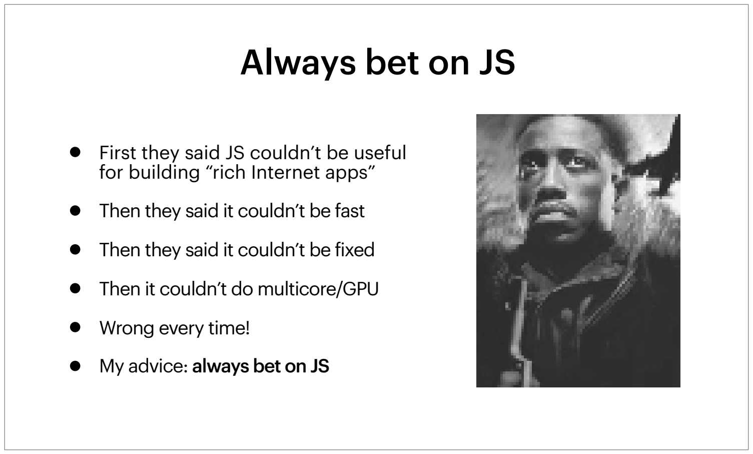
It was a trend Eich had seen coming. His 2011 talk concluded: "Always bet on JavaScript."[16] Over the next decade, that line became a catchphrase.

React popularizes the component model

In 2013, a team inside Facebook open-sourced React, a UI framework that they were using for Instagram.

Where existing frameworks like Angular grouped code together by what it did. React grouped code together by where it lived on the web page, mixing HTML, CSS, and JavaScript in a component-based format called JSX.

When users filed out form fields, toggled buttons on the page, or the browser received new data from a backend server,



Always bet on JS

- First they said JS couldn't be useful for building "rich Internet apps"
- Then they said it couldn't be fast
- Then they said it couldn't be fixed
- Then it couldn't do multicore/GPU
- Wrong every time!
- My advice: **always bet on JS**

[16] **Brendan Eich presentation**
September 2011, CapitolJS

React would update any affected components, without having to re-render the whole page.

Built on an engine known as the "virtual DOM," React was a tremendously clever piece of software. It borrowed concepts from software fields as disparate as compilers, functional programming, and video game graphics. The end result was a framework that allowed developers to define their application as modular pieces of user interface that could be composed together.

It took a couple years for developers' initial incredulity to wear off. But, backed by a dedicated Facebook engineering team, React kept getting better, and it kept gathering steam.

In some ways, React's component model was the perfect match for the mobile era. For designers in the desktop era, a web page was like a magazine spread. Visual elements could stretch across a page – infographics, full-width Flash animations, dotted lines weaving around – everything was fair game. But in the mobile era, designers had to think about how their design would play on multiple form factors – smartphone, tablet, desktop.

A typical desktop browser window was 900 pixels wide. So designers started to think about how to tell a story with three elements that were 300 pixels wide, rather than one element that was 900 pixels wide. They could sit side-by-side on desktop, but flow top to bottom on mobile. The web was moving from a *canvas* to a *component tree*. [18]

By 2019, React had surpassed Angular.[17] React ushered in the component era of web development, inspiring later frameworks like Vue and Svelte, also based on the component model. Developers quipped that React was like Legos.

TECH FEATURE

From design systems to component libraries

The idea of design systems had existed since the mid-2000s: a style guide to help maintain a consistent look and feel throughout a company's web properties. Unfortunately, no matter how well-written, rulebooks don't enforce themselves.

But by making it possible to write HTML, CSS, and JavaScript in a single file, React allowed design systems to be implemented in code – in component libraries. Instead of telling developers, "write headers that follow the following guidelines," designers could say, "use this header component."

In addition to components, design systems also encoded other web design elements like icons, animation, illustration, and layout; by standardizing these across organizations, they freed designers to compose with visual metaphors.

From 2016 to 2018, design systems spread, starting in large technology companies like Salesforce, Github, and Airbnb, and spreading to consumer companies like Audi and Capital One.

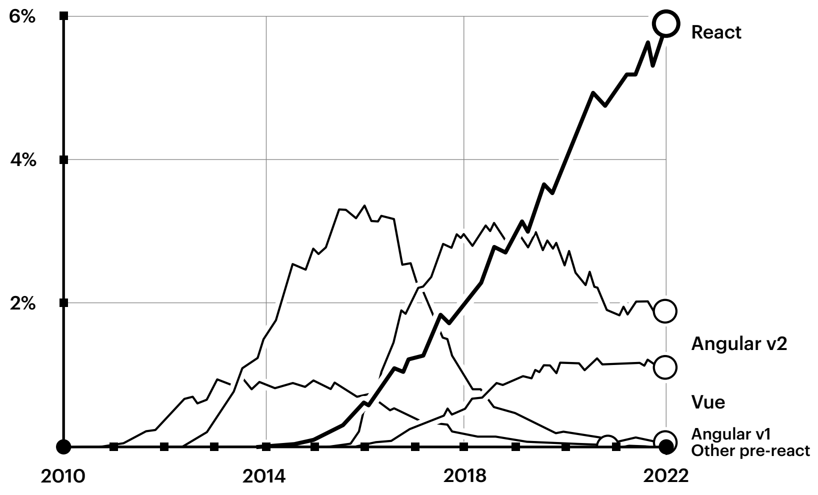
Frontend teams divide and conquer

The component model also re-organized how frontend teams worked.

Component trees (with React) also allowed a page to be broken into sections, and ownership split between different developers or squads. Code written by one group couldn't break code written by another group.

Component libraries (via tools like Storybook) divided work similarly, allowing frontend developers with a strong sense for design to create general components, and other developers to focus on stitching those components together into user experiences. These tools served as the *yin* to React's *yang*, adding *discoverability* to *modularity* and *componentization*.

React allowed larger technology companies to focus on key



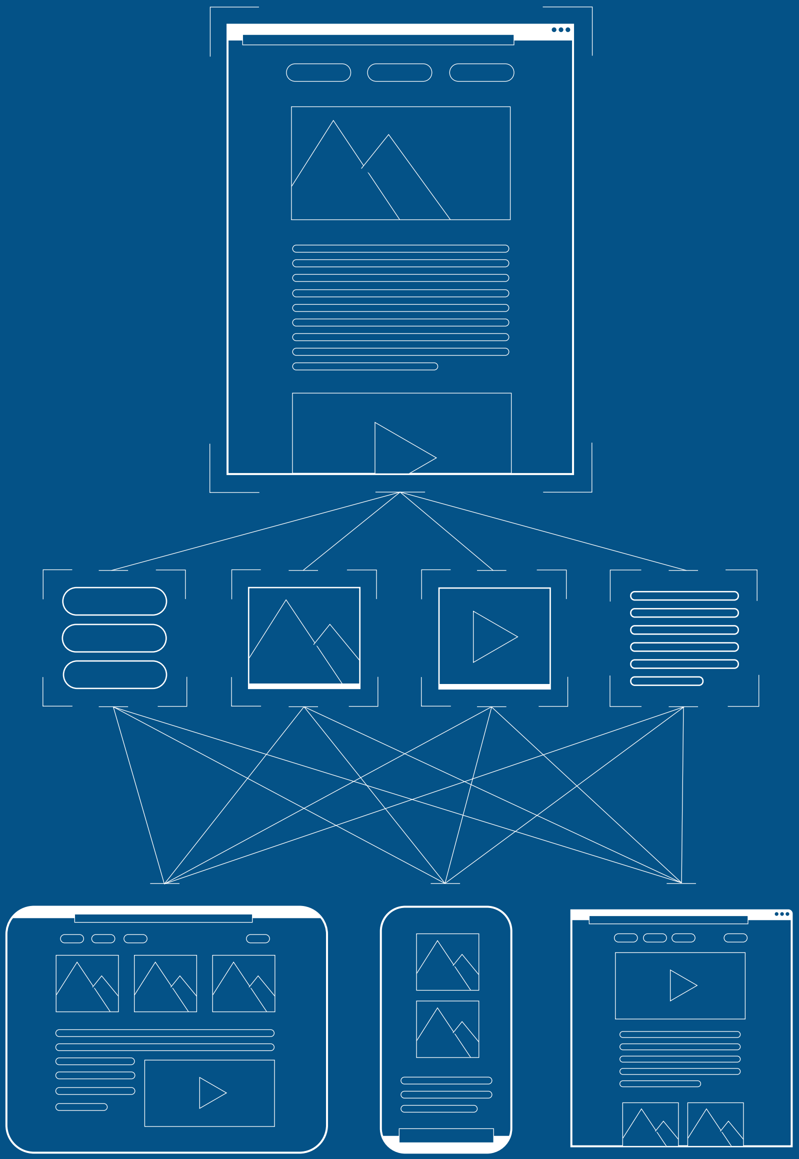
17 **JavaScript framework popularity**
% of StackOverflow questions tagged

elements of user experience with a previously unknown intensity. At Airbnb, engineer Maja Wichrowska created and open-sourced the company's date picker. Then, over two years, she wrote over 350 commits and 20,000 lines of code, adding dozens of configuration options at the request of different teams.

Key takeaways

Back in the 00s, JavaScript wasn't very good, allowing languages and frameworks like Ruby on Rails, ASP.NET, and PHP to gain traction. But over the last decade, JavaScript first caught up – and then blew by – these languages.

Today, JavaScript is the most widely used language for web development, along with React as a component framework.



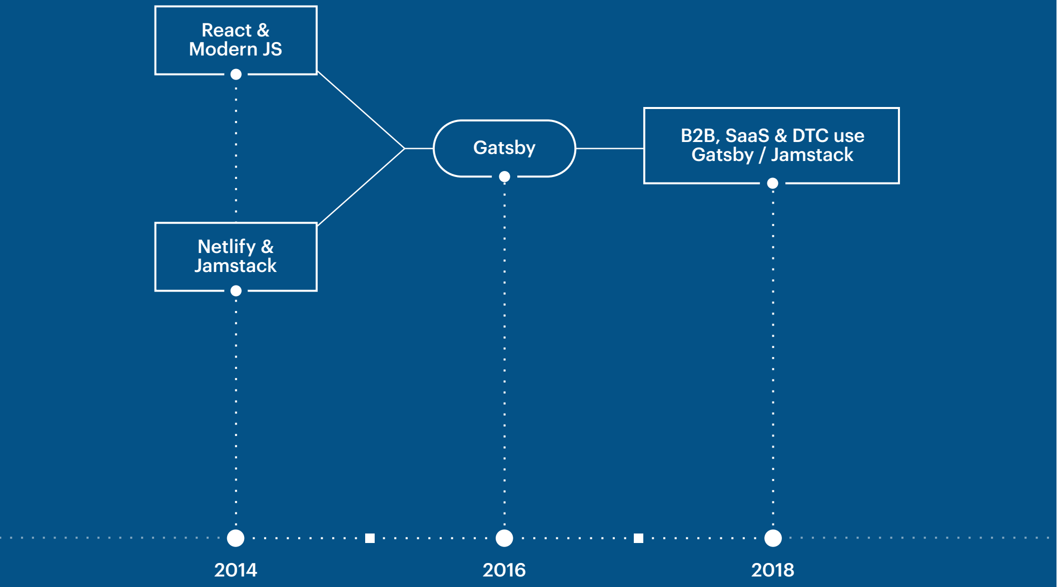
18 From canvas to component tree

MODERN DEV

CHAPTER 7

GATSBY AND THE JAMSTACK

THE BIRTH OF GATSBY • NETLIFY & THE JAMSTACK • EARLY ADOPTERS



This is where I come into the story.

In 2015, I was a developer in Silicon Valley. I'd worked for a couple of different SaaS companies, watching the cloud expansion play out. I'd started on the business side, but two years earlier with the encouragement of my close friend Kyle Mathews I transitioned into programming.

Since then, our conversations frequently revolved around programming tools. In 2015, Kyle told me about a weeklong project he was working on: rebuilding his company's website. He had fallen in love with React, convinced that components were the future of the web.

A tinkerer by nature, Kyle decided to build a React-based static site generator. He open-sourced the framework, and following the literary tradition of the space (Jekyll, Hugo), he called it *Gatsby*. Most of the first Gatsby sites were deployed on Netlify, a service for hosting static sites.

"All fast websites are alike..."

Site speed challenges on mobile had fostered the growth of a web performance community. There were a mix of pragmatists, focused on the conversion benefits of site speed, and idealists, who saw poor site performance as an accessibility issue for developing regions.

Websites should be fast, these developers believed, not just on fast Wi-Fi or 4G connections, but on 2G or slow 3G. The web shouldn't just work well on iPhones or laptops, but on budget Android phones throughout the world. These developers created a set of sometimes esoteric best practices for improving site speed: asynchronous Javascript loading; optimized web fonts; tree-shaking; code-splitting; compression algorithms like gzip and brotli. (Unlike in the popular HBO sitcom *Silicon Valley*, new compression algorithms did not spawn new startups).

Some focused on evangelism, publishing lists of these practices in outlets like *Smashing Magazine*. As part of this community, Kyle took a complementary path: he made many of these practices, like progressive image loading and preloading links, the default in Gatsby.

“All fast websites are alike, but all slow websites are slow in different ways,” Kyle joked, paraphrasing Tolstoy. As he added more of these features to Gatsby, he’d hear people remark about how *shockingly* fast Gatsby websites were.

Integrating with headless CMSs

Gatsby, like earlier static site generators, let developers compose content by writing text files in Markdown and saving them in their codebase. This worked for certain types of content, like developer blogs, or code documentation.

But Kyle thought it should do more. He had spent years in the Drupal community, full of organizations like universities with dozens or hundreds of content writers and complex content schemas. Watching API-based “headless” Drupal and purely headless CMSs like Contentful, Kyle saw the possibility of Gatsby to move beyond “static” developers’ websites.

Kyle saw a new architecture, with marketers in a headless CMS writing content, seamlessly integrated with developers in Gatsby building components, fetching data, and creating page templates. He began to build CMS integrations into Gatsby, adding integrations for Contentful, WordPress, and other CMSs.

As Kyle got more and more into this idea, he quit his job, paying his San Francisco rent with consulting gigs. Kyle and I would take lunchtime walks at the popular Mission Dolores Park

and talk about Gatsby. Eight months and forty alpha releases later, Gatsby v1.0 was finally ready to be launched into the world.

By the next week, the relief of the launch had turned into frenetic energy. Rather than writing code, Kyle was now spending most of his time reviewing Gatsby code that other people were writing. “I’m basically the tech lead now,” he laughed. The launch had created a huge surge of interest in Gatsby, and people were playing around with the new APIs by building plugins and contributing them to the community.

Software companies want React websites

With React growing, a few conferences invited Kyle to speak about Gatsby. When he asked the audience who’d used Gatsby, he was shocked when hundreds of developers raised their hands. By 2017, hundreds or thousands of B2B SaaS companies had migrated their SaaS applications to React, hired or trained a team competent in building React frontends, and created lots of components that embodied the company’s visual metaphors and branding.

As these companies grew, they inevitably decided to redesign their website. Often, they turned to the React frontend developers they’d hired. Armed with a set of components and a belief in component-based UI, these developers wanted a website framework that let them build with React and the modern JavaScript ecosystem. So they turned to Gatsby and what was beginning to be called the Jamstack.

Over 2017 and 2018, a number of high-profile “unicorn” startups – design tool Figma, hosting service DigitalOcean, email platform SendGrid – migrated their homepages to Gatsby.

Marketing teams streamline content creation

In addition to being a natural tool for the many React SaaS web app companies, the Jamstack and a modular website architecture let many tech marketing departments practice the gospel Silicon Valley was preaching: well-integrated, best-of-breed tools working together to streamline and optimize business processes.

One example is Housecall Pro, an LA-based scheduling and dispatching SaaS for home services companies like plumbers or HVAC installers. To raise awareness and increase leads generation, they considered a SEO campaign targeted to the fifteen industries they served.

Housecall Pro wanted to create industry-specific landing pages. These would highlight product features: scheduling, invoicing, and estimating. Or they would customize gated content: cold call, proposal, and quote templates.

Multiplying the number of industries by the number of templates, this would entail creating hundreds of pages. Housecall Pro set up their campaign by creating templates in Gatsby, so that a content team could build new landing pages by creating content objects in Contentful. Before they connected Gatsby to Contentful, they were launching a page or two a day; afterwards, they were creating dozens of pages a day. The campaign increased Housecall Pro's SEO traffic by a factor of 10.

The new Jamstack architecture

With the growth of Gatsby, as well as the emergence of other React- and Vue-based frameworks like Next.js and Nuxt.js, the static site scene started to change. People began to talk about the new JavaScript-driven "Jamstack" as an alternative to

the old PHP-driven "Lampstack".

The phrase "Jamstack" captured something essential. This was a fundamentally new architecture. Jamstack was once primarily used by simple sites like blogs. But now with React, Jamstack sites had interactivity. Nor was the Jamstack limited to developers: they could connect to headless CMSs, empowering marketers.[19]

Jamstack sites were *fast*: serving HTML from a globally distributed CDN is faster than running servers and databases. Jamstack was *secure*: there's no attack surface. Jamstack was *scalable* – backed by a CDN, sites wouldn't go down in a burst of traffic. Developers loved Jamstack, feeling that the architecture was right. Demand gen marketers loved *site speed* and optimized mobile performance. Content strategists loved headless CMS flexibility.

As Gatsby and the Jamstack became known for fast websites, they began to get pulled into larger and larger web projects.

DTC companies get excited about performance

We also started to see a lot of excitement around performance from DTC e-commerce.

E-commerce marketing executives typically track conversion as a high-level company metric. After site launch they reported staggering jumps due to fresh designs and faster page loads. The men's jewelry brand Jaxxon saw their conversion rate double following a three-month migration.

Soon, a number of name-brand companies began jumping on Gatsby and Jamstack. The health company Ritual; the razor brand Harry's; the meat alternative company Impossible Foods.

Each project was fairly customized, starting with some sort of an architecture diagram including a number of different systems, where each bit of data lived and how it synced. As such, they involved relatively talented developers and designers, as well as larger budgets than a more "off-the-shelf" Shopify site.

The projects were more complex than the average e-commerce build. They required selecting different systems, drawing an architecture diagram, and figuring out data flows. They needed a certain type of developer or agency. But they proved ROI in spades.

The challenges of building an e-commerce site -- persisting cart state across pages, fetching and filtering products, purchasing items -- are the challenges of building an "application". By combining Jamstack's performance and React's interactivity, an e-commerce backend, headless content management, and new analytics tools, these companies built a powerful new engine for online business.

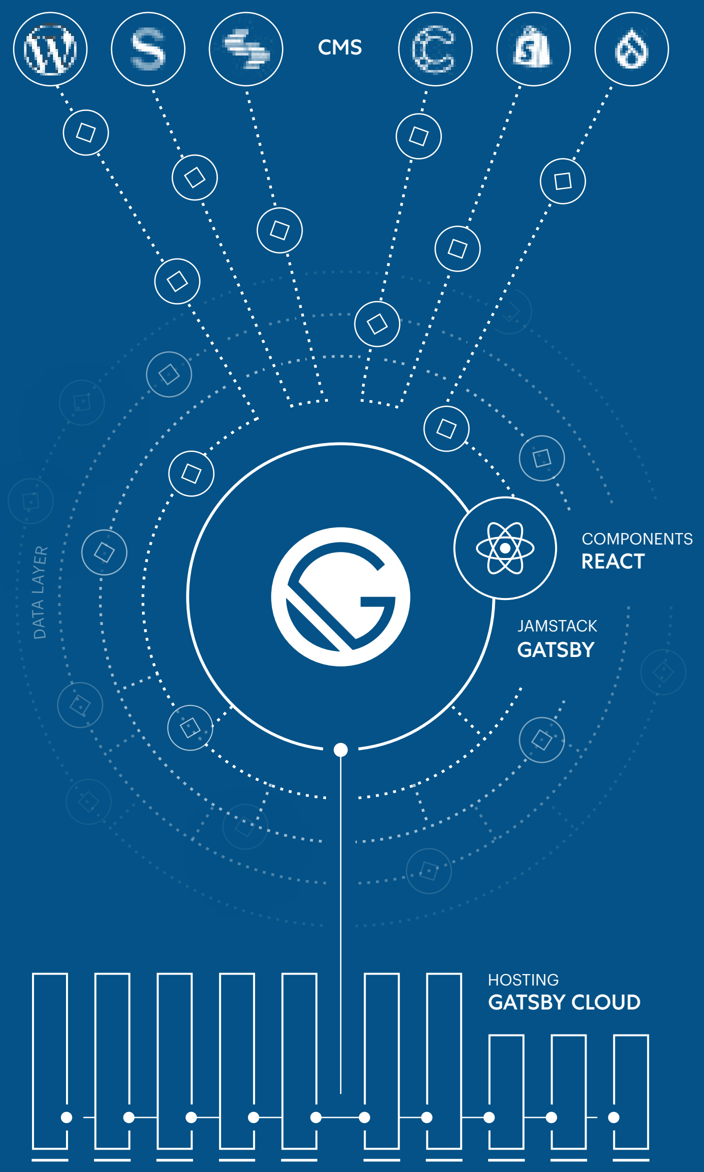
Key takeaways

The Jamstack architecture, led by Gatsby and other frameworks, enabled websites to be fast, scalable, and secure.

When integrated with headless CMSs like Contentful, they formed a new stack for the content web.

Technology companies began adopting this stack because their developers loved React.

DTC e-commerce companies began using this stack because they were excited about performance.

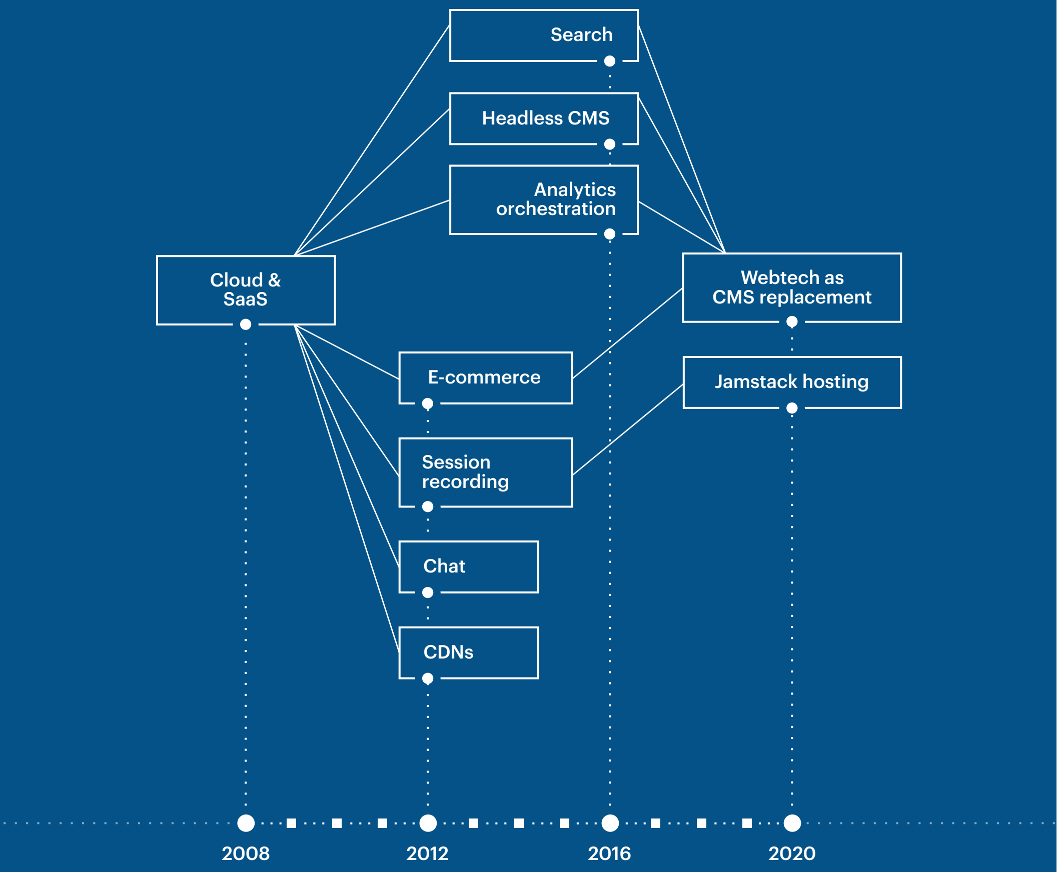


19 Gatsby website architecture diagram

CHAPTER 8

THE WEBTECH SOLAR SYSTEM

HEADLESS CMS • JAMSTACK HOSTING • WEB SERVICES • MARKETING ANALYTICS



In the early 2010s, a host of cloud-based SaaS tools had sprung up (see Chapter 4). By the late 2010s, this “webtech” had become deeply embedded in every website-related role:

→ **Content teams** began writing and editing in headless CMSs.
→ **Developers** used new Jamstack deploy and hosting tools, and plugged in services like search and auth.
→ **Demand generation marketers** used fine-grained analytics tools.
→ **Product and design teams** used session recording to understand visitors’ individual experiences.
→ **Marketing operations** used analytics orchestration tooling to manage data flow with the explosion of new tooling.
→ And **e-commerce companies** brought a whole suite of tools from pricing and promotion to subscriptions and refunds.

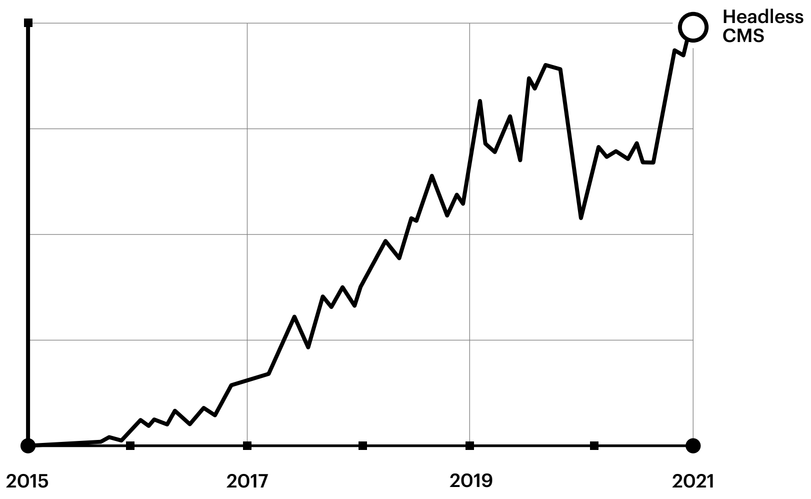
Operated on the SaaS model, these tools were relatively affordable, easy to implement, and up to date with the latest developer trends. Their modularity meant teams could choose which tools and insights they needed and plug them into their workflow.

Headless CMS: reusable, flexible content

By 2015 or 2016, the idea of a "headless CMS" was emerging. Headless CMSs combined *schema creation* and *content entry*, allowing content teams to update and structure the data that fed into the site. They then exposed that content via an API. They were called headless because they lacked web page building, or site hosting, functionality, which meant that they could be used with custom frontends. (20)

In addition to pure headless CMSs, large open-source CMSs – WordPress, Drupal – added APIs, as did some enterprise CMS vendors.

The headless CMS wasn't specific to any programming language; you could use it with Java or C# to build an in-house enterprise application, Node.js to build a public-facing SaaS,



20 **Google trends: Headless CMS**
Search traffic over time

or Swift to build an iOS app. Modularity enabled flexibility. A headless CMS was less the backbone of a specific website, and more a piece of an overall architecture for multiple applications.

In a 2016 blog post on headless, CMS guru Deane Barker highlighted use-cases like *content aggregation* and *content middleware*, putting together content from multiple sources to power several different channels. Headless CMSs were also bolted on to existing systems. A banking website processed forms through its main application, but stored help-text on form fields in a headless CMS, where it could be edited by a documentation team.

The concept of a CMS was changing, from a linear pipeline to the website, to more of a core powering a network of views/instances/satellites, and being worked on by multiple teams.

CATEGORY	DESCRIPTION	WHY IT MATTERS	PRODUCTS
Headless CMS	Lets marketers write/edit; exposes content via API.	Structured content is the core of most websites.	↘ Contentful ↘ DatoCMS ↘ Contentstack ↘ Prismic ↘ WordPress ↘ Drupal

Build & deploy platforms: hosting for Jamstack

Simple hosting for traditional applications is a Heroku node or DigitalOcean droplet – a single container with access to the public Internet. But Jamstack sites were different. Rather than armies of individually configured servers, they deployed to a CDN.

Early on, Netlify was the main Jamstack host, but a new wave of services soon emerged, including both larger cloud players (AWS Amplify, Cloudflare Pages), and startups optimized for one specific framework: Gatsby Cloud for Gatsby, and Vercel for Next.js. Each of these services allowed developers to add environment variables; they rebuilt the site when new code was pushed to Github or other version control systems. When developers proposed code changes via a "pull request", services would create a version of the new site in staging, known as a deploy preview, to enable collaboration and feedback.

Cloud platforms allowed frameworks to support larger sites while reducing build times to minutes. Performance optimizations made the Jamstack viable for sites with tens or hundreds of thousands of pages.

CATEGORY	DESCRIPTION	WHY IT MATTERS	PRODUCTS
Deploy & hosting platform	Hosting; re-deploys when code/ content change	Makes Jamstack easy.	↘ Gatsby Cloud ↘ Vercel ↘ Netlify ↘ AWS Amplify

Search and auth: drop-in dynamic services

Back in 2015, web search was hard. Open-source tools like Elastic needed a serious backend lift. And "do it yourself" involved developers writing ranking algorithms, handling pluralization, word variants, typos…

In 2014, two French search engineers built and launched Algolia as an API-based search for websites. It was configurable, but worked out of the box. Enterprise search gurus colored themselves impressed. "You can, with a good team, build just about any searchy thing with [other technologies]," blogged Doug Turnbull, an expert on the search library Lucene. "But it does take a team." Algolia especially shined in more complex use-cases – "faceted" search including product category or price filters on

e-commerce sites; grouping results by site section, and so on.

A similar plot unfolded in authentication. Auth was simple to build but quickly grew complicated. Social login via Gmail or Facebook. Managed OAuth flows. Magic links. Single-sign-on. SAML or Active Directory support. The Auth0 founders wrote a book on authentication, and then decided to build a service to manage the complexity for developers.

CATEGORY	DESCRIPTION	WHY IT MATTERS	PRODUCTS
Search	Index site content; give best match for queries	Better product discovery, reduced bounces	↘ Algolia
Auth	Handles auth, social login, SSO, PW reset, etc	Easier user and account functionality	↘ Auth0

Advanced analytics: funnels, cohorts, conversions

Google Analytics launched in 2005. It is the most universal web service around. But web analytics has changed a lot in the last decade and a half. And increasingly, teams reach for more.

Google Analytics provides aggregated statistics like page views and ad attribution. But with the growth of mobile and demand generation, marketers were trying to get a fine-grained view of individual visitors. Growth marketers needed to understand how to value *individual visitors* so they could optimize metrics like Customer Acquisition Cost (CAC) and Customer Lifetime Value (LTV).

To value individual visitors, marketers needed a rich picture of user intent. They needed to correlate actions – browsing, purchase, search history – with IP-based demographic information and lead source. They needed to create funnels, segment users, and create cohorts. And to do this, marketers needed a new generation of tools, built around event-based data.

CATEGORY	DESCRIPTION	WHY IT MATTERS	PRODUCTS
Event-based analytics	Build funnels, segment users, track conversion.	Faster, cheaper customer acquisition.	↘ Heap ↘ Amplitude ↘ Mixpanel

Session recording: replaying visitors' experiences

While quantitative tools helped marketers analyze user behavior, other website roles sought greater visibility into individual user experiences. Session replay evolved to record (and replay) a user's actions on a webpage, like scrolling or clicking.

Developers would use session recordings while debugging; they were often more intuitive than error logs. Designers and UX researchers would use session recording to browse users' experiences in addition to (or instead of) setting up user interviews. Customer support teams would use session recordings to better understand the context behind customers' questions.

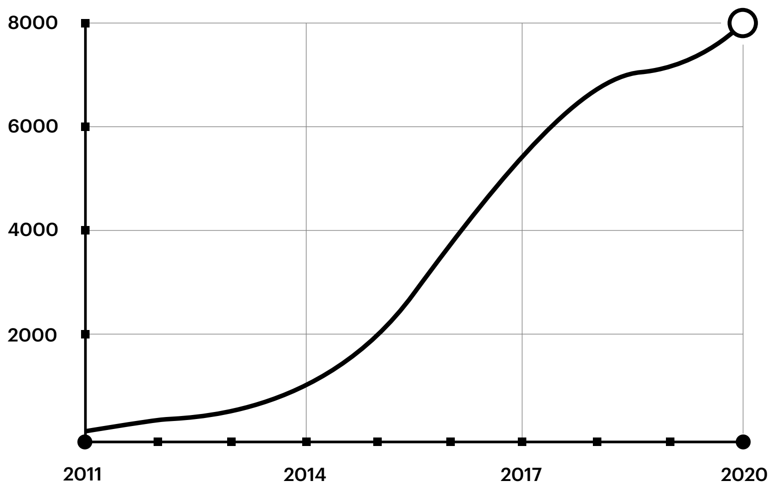
CATEGORY	DESCRIPTION	WHY IT MATTERS	PRODUCTS
Session recording	Watch individual user interactions; view heatmaps and rage-clicked elements.	Walk a mile in your visitors' shoes.	↘ Hotjar ↘ FullStory

The CDP: orchestrating marketing data

Event-based analytics and session recording tools were far from the only kind of new marketing tool; in fact, perhaps nowhere was the SaaS explosion more apparent than in marketing technology. There were 150 "martech" tools in 2011 – a number that doubled every year for five years, reaching 4,000 by 2016 and 8,000 by 2020. [21]

While marketers appreciated the additional functionality and finer-grained analytics, managing multiple services added complexity. One study found that, over three years, a typical marketing team tried out ten to twenty different services.

Some used Google Tag Manager, which helped somewhat by making it easy to add new services. But customer data platforms (CDPs) took that further, allowing teams to replay historical



21 **Growth of the martech landscape**

data into new services, or try multiple services simultaneously.

"[Most marketing teams] start with an analytics use case," wrote Calvin French-Owen, co-founder of the CDP Segment, after analyzing the company's data. "From there, they expand to email, screen recording, and live chat tools, and then to more specialized use cases like advertising and raw data."

Because CDPs had the ability to send data to multiple services, they enabled teams to replace multiple analytics trackers with a single tracker, boosting web performance. This made performance-focused frontend developers happy, and solved a key problem created by the marketing technology explosion.

CATEGORY	DESCRIPTION	WHY IT MATTERS	PRODUCTS
Customer data platform (CDP)	One tracker to send data to multiple services.	Unify fragmented marketing analytics data.	↘ Segment ↘ Rudderstack

The new "solar system" web architecture

By the 2020s, an early-adopter B2C company would be using several of these tools, loosely integrated together. This is known as a federated architecture, which then-Contentful CMO Bridget Perry compared to a solar system.

Consider a mid-sized European online retailer. They spend the most time thinking about e-commerce. Let's call that Jupiter, with reviews, rewards, logistics, and other auxiliary commerce systems its moons. Their build and deploy platform is Saturn; its rings and moons are their Jamstack and Javascript frameworks, dozens of React components, a search service, a testing platform, and so on.

Earth is their headless content management system, with built-in content permissioning and approval workflows, orbited

by their internationalization workflow. The asteroid belt are their various analytics tools. [22]

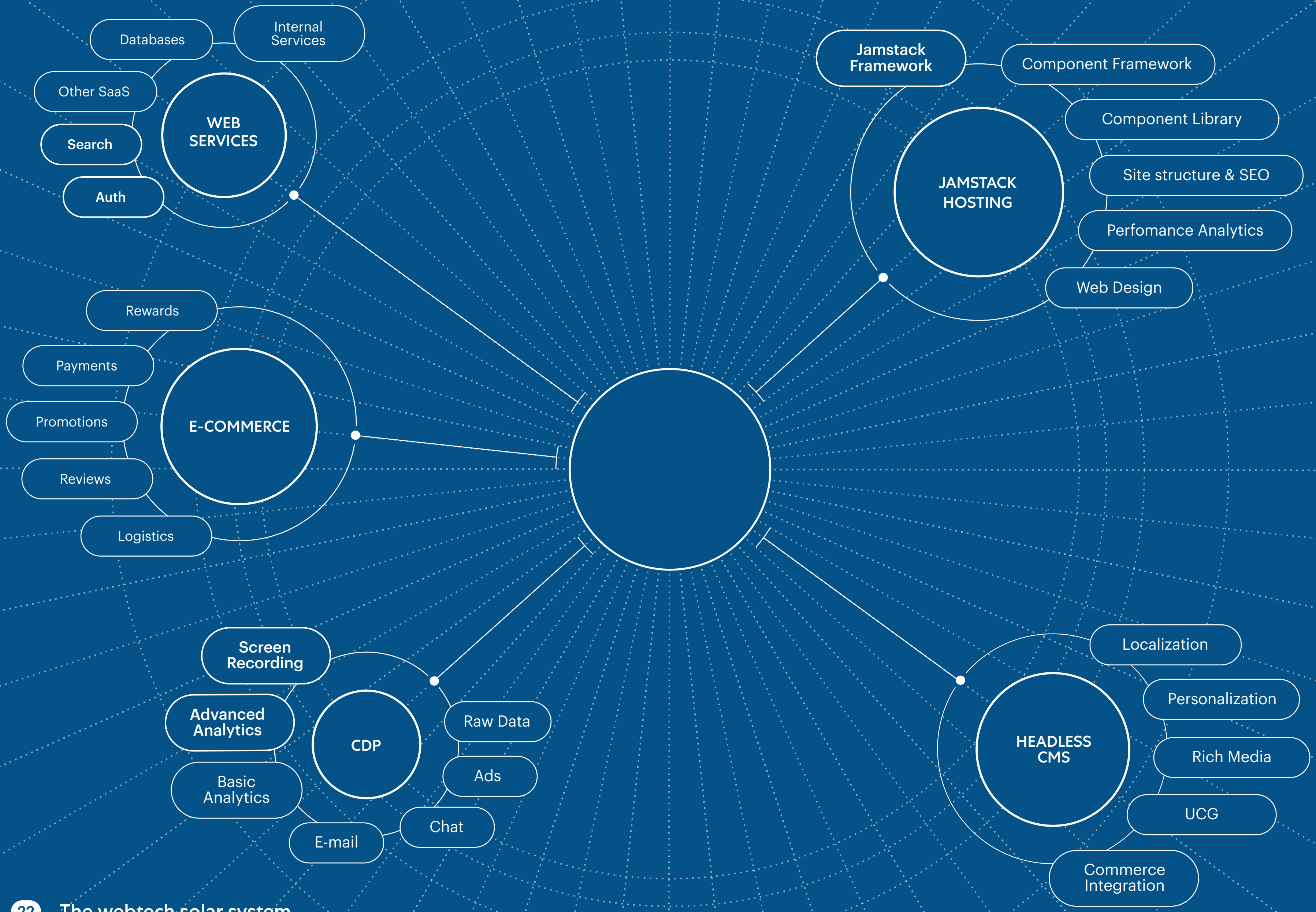
Put together, a "solar system" composed of these "planets" made a compelling alternative to the traditional CMS.

Key takeaways

By the late 2010s, "webtech" cloud-based SaaS had become embedded in every role involved in building and operating websites.

This included headless CMSs and Jamstack deploy and hosting tools, to services like search and auth; to fine-grained analytics, session recording, and analytics orchestration; to e-commerce platforms and tools.

From segmenting and tracking conversion, to better understanding visitor experiences, to equipping developers with modern tools, to optimizing conversion, this stack offered significant benefits over traditional, monolithic CMS websites.

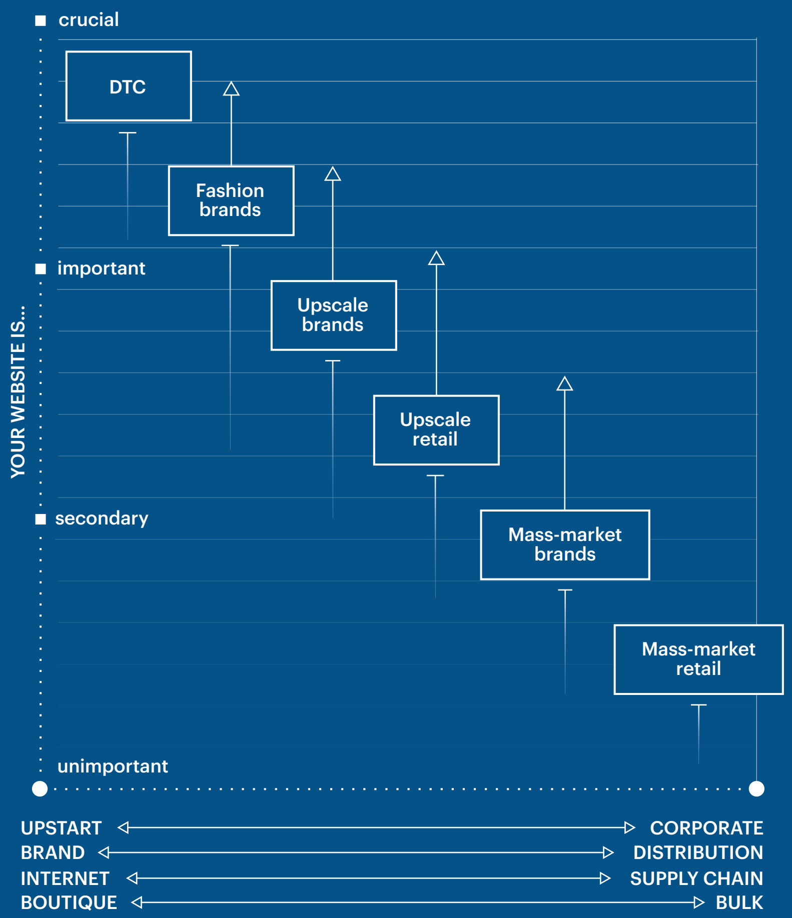

Databases
Internal Services
Other SaaS
WEB SERVICES
Search
Auth
Jamstack Framework
Component Framework
Component Library
JAMSTACK HOSTING
Site structure & SEO
Perfomance Analytics
Web Design
Rewards
Payments
Promotions
E-COMMERCE
Reviews
Logistics
Screen Recording
Advanced Analytics
CDP
Raw Data
Ads
Basic Analytics
E-mail
Chat
Localization
Personalization
HEADLESS CMS
Rich Media
UCG
Commerce Integration

E-COMMERCE

CHAPTER 9

BRANDS AND RETAIL GO DIGITAL

HEADLESS CMS • JAMSTACK HOSTING • WEB SERVICES • MARKETING ANALYTICS



In the mid-2010s, "e-commerce" was mostly online-first stores like Amazon, or online-first brands. Retail commerce was almost entirely in person, with a credit card swipe at the end. But around 2017 or 2018, that started to change.

The e-commerce ecosystem explodes

One cause: e-commerce systems, catalyzed by hundreds of major DTC brands and hundreds of thousands of Shopify stores, were getting better and more complete. Software and services spawned to handle all the complexities of running a storefront. In many ways, it paralleled – and recreated – the brick and mortar world.

Retail built parking lots; e-commerce invested in logistics networks. Retail employed greeters; e-commerce used full-screen email capture modals.

Retail maintained stockrooms; e-commerce needed inventory management. Each handled product variants – "do you have that shoe in a size 10.5?"

Retailers mailed out circulars with discounts; installed clearance racks; issued store cards, and offered bulk-buy discounts ("buy two get one free"). E-commerce, too, developed rewards, promotions, loyalty and affiliate programs. On top of these, they added a host of online-only tools: reviews, email automation; customer video testimonials; buy now pay later; cart abandonment, push notifications, SMS, chatbots. The race was on to grab customers' attention and hold onto it by any means possible.

Shopify's app store facilitated this explosion. Merchants started with a working base and gradually added apps as upgrades; developers could go indie, build a very focused app, and make a living. And because apps ran in the cloud, many

TECH FEATURE

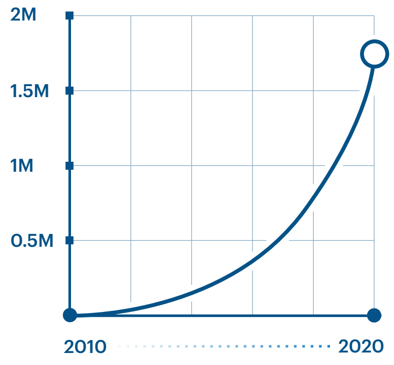
Shopify

Shopify was created by German-born programmer Tobi Lutke in 2004 when Lutke couldn't get the design for his snowboard store to work on an existing platform. In 2009, when Shopify had 5,000 merchants, it launched an app marketplace with features like invoicing, reviews, cart abandonment, SEO, and so on.

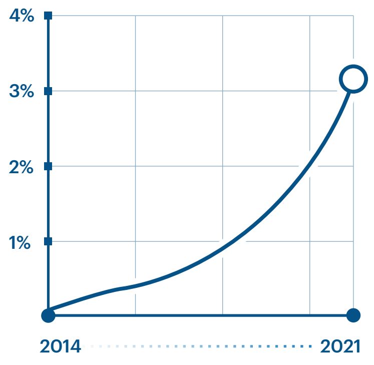
Shopify's ease of use, customizability, and feature-richness drove growth, tripling merchants every two years for a decade until it reached 1 million in 2019. By this point, Shopify had become the most popular e-commerce system on the planet.

Shopify's cloud-based App Store helped bootstrap the modular commerce ecosystem. Shopify encouraged apps to run in the cloud; WordPress plugins, by contrast, ran on a WordPress instance.

That meant that with a little tweaking, most Shopify apps could be used outside the ecosystem – especially after 2021, when "headless mode" became a requirement to work with Hydrogen, Shopify's new Jamstack framework.



23 Shopify merchants



24 Shopify as % of web

popular services built initially for Shopify later added support for other platforms.

CATEGORY	DESCRIPTION	WHY IT MATTERS	PRODUCTS
E-commerce platform	Coordinates payment, logistics, finance, etc	Manage complexity of online business	↘ Shopify ↘ BigCommer ↘ CommerceLayer

Incumbent consumer brands shift online

DTC companies concentrated in certain kinds of consumer goods: shoes, jewelry and watches, supplements, skincare, cosmetics, women's sleepwear, mattresses.

Their playbook worked for certain types of personal items, where customers did some research before buying (known as "considered purchases"). It especially worked when products could be positioned around luxury, self-indulgence, or personal improvement.

Incumbent brands in these spaces reacted accordingly.

Nike started selling online way back in the late 90s. But in 2014, e-commerce was only 3% of Nike's overall revenue. But that changed quickly, growing to 10% of revenue by 2018, and passing 20% in 2020.

Mattress manufacturer Tempur Sealy's e-commerce sales crossed 5% of revenue in 2016 and 10% in 2018. In 2020, Tempur Sealy made more money selling mattresses online – about $500 million – than their public DTC competitor Casper made in total.

When brand executives reported these numbers on quarterly earnings calls, analysts cheered. Online sales had a higher margin than sales through retailers; there was no middleman.

Of course, this strategy only worked for certain types of

WEBSITE PROFILE

Strivectin.com

Founded in 2002, StriVectin is a skincare brand with lines of anti-aging products sold both through cosmetics retail channels as well as online direct-to-consumer. With a base of fans making regular purchases, their e-commerce store was quite popular.

Over time, it was becoming increasingly difficult for the team to make changes to their seven-year-old codebase, built on a legacy platform (Magento). So in 2018, Strivectin decided it was time for a switch. They engaged an agency, Elevar, with experience in the modular web. The new stack: Shopify for e-commerce, Prismic for headless CMS, Gatsby for their Jamstack framework.

Elevar did the project in stages – first building out all of their components in React and Storybook, then building out the pages in Gatsby, then adding supplementary e-commerce functionality: Recharge for subscriptions; Yotpo for reviews, and Swell for rewards and referrals.

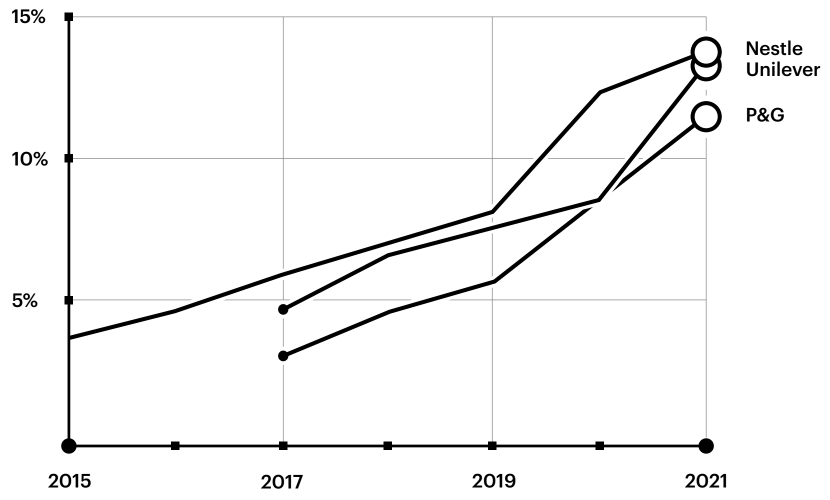
The flexibility allowed them to create custom, brand-centric display modules, such as before-and-after photos for each product. Using React components helped Strivectin give the site a unique look and feel. And Gatsby helped boost performance and conversions. The transition was seamless.

"This has been the smoothest cut-over of my entire career," said Felicity Sissner, Strivectin's chief digital officer.

goods, where direct-to-consumer brands thrived – where products could be positioned as luxury, and "considered purchases" where consumers did some amount of research before buying.

Consumer goods conglomerates chase growth

In a few arenas – coffee, razors, pet food – the large consumer goods conglomerates competed against DTC upstarts, and started to respond in kind. In others, they saw online as less an opportunity to *sell* to consumers and more an opportunity to *learn* about them. Companies like Tyson and Unilever launched content sites with food recipes and home cleaning tips. (25)



25 **CPGs go digital**
Online Revenue as % of total

WEBSITE PROFILE

Little Caesars

Started in 1959, Detroit-headquartered Little Caesars has grown into the third-largest pizza chain in the world. In January 2020, the company was excitedly preparing for Little Caesars' first-ever Super Bowl ad, starring Rainn Wilson from The Office.

The web team was hard at work on a website relaunch that allowed customers to order pizza online on the website for pickup or delivery. They chose Gatsby and the Jamstack because they loved React, and because they wanted to build a fast, secure site that wouldn't go down under a surge of traffic.

The site was fully operational a mere three days before the Super Bowl, and the ad's premiere. "The marketing team was super stressed," recalls Andrew Smith, the project's architect. "They definitely needed the new site ready for the big day, but [our stack] enabled us to make the tight deadline."

And the new site handled Super Bowl traffic without a hiccup. "Traffic spiked immediately after the ad aired. We had 3,000 simultaneous users, then 10,000, then a few seconds later it was 15,000," Smith says. And the team didn't have to worry that the site would go down. "Because we built with Gatsby, we knew we had the performance to handle the traffic without even thinking about it."

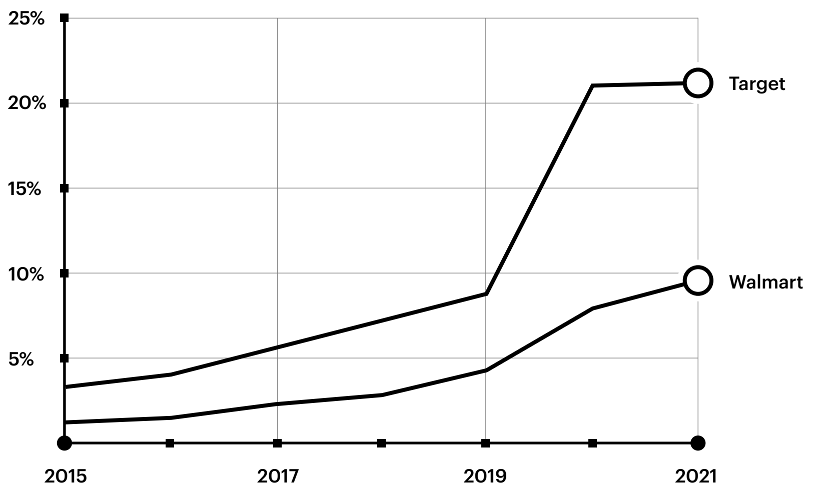
The relaunch timing was fortuitous. Shortly afterwards, as COVID came on the horizon, Smith and the team received an emergency request from international business leadership to add e-commerce capabilities in Canada.

They delivered the requested functionality in under a week.

And for their portfolios of other, more pedestrian consumer goods – toilet paper, soft drinks, socks – these conglomerates began launching online "B2B2C" efforts, trying to shift their relationships with distributors online.

Retail's COVID Shockwave

When the world shut down in March 2020, retail was hit hardest. Online shopping jumped to the top of the priority list for every Western retailer. In 2019, only 4% of the largest 500 retailers offered online ordering with curbside pickup. By summer 2020, with the COVID pandemic in full swing, that number had jumped to 44%.(26)



26 **Retail adjusts for COVID-19**
Online Revenue as % of total

For retail stores, digitization required rethinking physical layout and staffing: adding a section to hold online orders; designating parking for curbside pickup; assigning staff to help these customers.

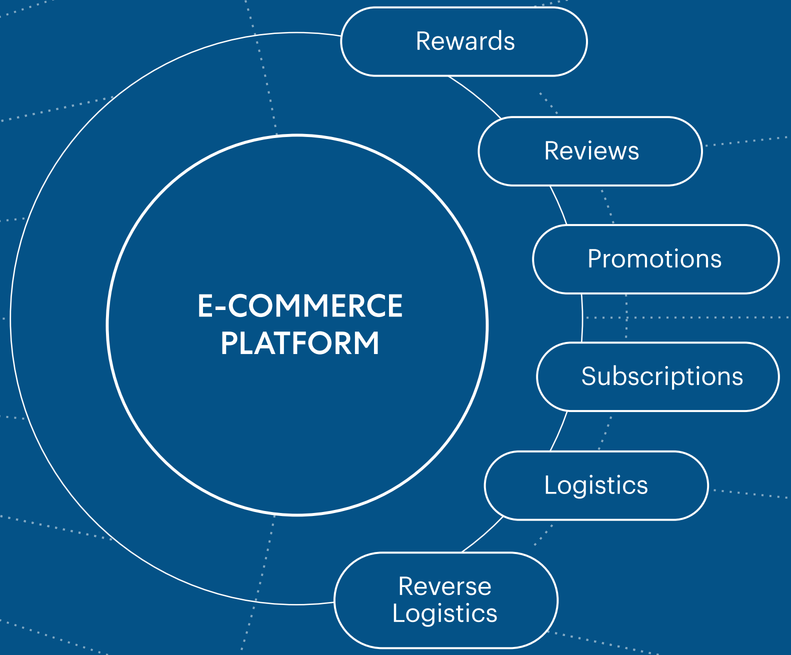
The transition was expensive. But the alternative – lose some proportion of traffic, or get "wrapped" by technology services like DoorDash or Instacart – was even more unappealing.

COVID brought brands and retail into the digital era. Commerce was converging toward the DTC form factor. And with consumer companies slowly becoming e-commerce companies, they were increasingly turning to the modular web.

Key takeaways

The growth of e-commerce spawned a whole ecosystem of modular software tools to move each retail workflow and function online.

Catalyzed by the pandemic, consumer brands, CPG conglomerates, and chain retailers alike began converging towards the DTC form factor.



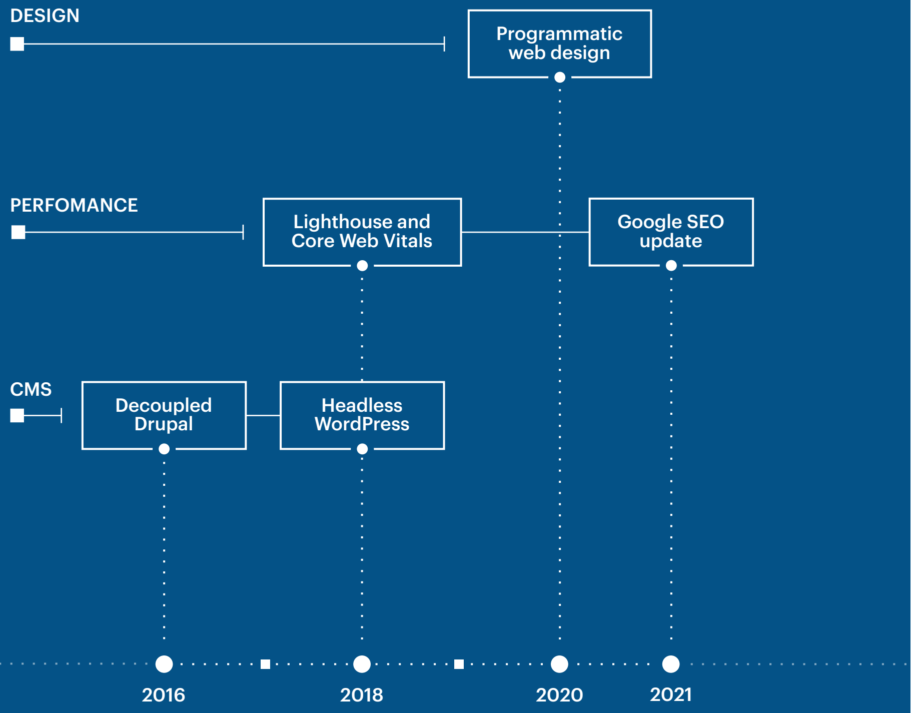
27 E-commerce ecosystem

MOBILE WEB CMS

CHAPTER 10

THE CONTENT WEB'S TIPPING POINT

THE CMS GOES HEADLESS • GOOGLE REWARDS MOBILE PERFORMANCE • WEB DESIGN GOES 3D



As the modular web matured, it meant different things to different people.

→ **Members of CMS communities like Drupal and WordPress** turned to the modular web as an alternative to a total migration.
→ **Companies valuing cutting-edge web design** turned to the modular web to build the aesthetics they wanted.
→ **Companies prioritizing SEO and conversions** turned to the modular web as Google rolled out new SEO updates prioritizing mobile performance.

CMS communities move toward the headless tipping point

Web development is an archipelago of CMS communities, each largely independent of each other and moving at different speeds toward the modular future.

HEADLESS ADOPTION	ENTERPRISE CMSs	WORDPRESS	DRUPAL
ENTHUSIASM	Low	Medium	High
API QUALITY	Low	Medium	Medium
BOUTIQUE AGENCY USAGE	Exploring	Experimenting	Doubling Down
ENTERPRISE USAGE		Exploring	Experimenting

WEBSITE PROFILE

CIA.gov

In the first week of 2021, the US's intelligence agency unveiled a new website, cia.gov, built on Gatsby, with content in Drupal.

"This wouldn't have been news if the site had stuck to the formal signifiers of government authority: dense bureaucratic text, link directories, declarative headers, nothing too fancy," wrote Ezra Marcus in the New York Times. "Instead, CIA.gov is set against a stark black background, offset by dots and lines that form topographical contours. There are subtle hallmarks of modern web design, like the site's animated scroll. The crisp lines and muted color palette suggest a minimalist branding strategy."

A spokeswoman for the agency commented that the design was intended to "pique the interest of talented applicants and provide a modern, relatable experience." A creative director commented that if he didn't read the copy, he would have thought this was the website of a "direct-to-consumer designer toothbrush [brand]."

They followed up a couple months later with a recruitment video featuring a 36-year-old Latina CIA officer talking about her experience as a "cisgender millenial" -- using words like "intersectional" and describing her experience as a person of color. The video went viral, triggering anger on right-wing media outlets.

Only the agency knows whether the campaign was effective in attracting applicants -- but it certainly did attract attention.

Drupal: the vanguard

Trusted as an open-source, highly configurable CMS, Drupal grew for much of the 2010s, expanding from its original base of open-source users to universities, nonprofits, and governments. It boasted a main vendor, Acquia, with hundreds of employees, selling into the Fortune 500.

In 2015 – long before "headless" started to gather hype – Drupal project founder and Acquia CTO Dries Buytaert outlined a reference architecture for decoupled Drupal. Buytaert had long been an advocate of the open web – and headless was one part of Acquia's strategy as it leveraged the headless trend to power customers' other digital properties.

The headless wave gathered even more stream in Drupal world in the late 2010s. Drupal crested as a proportion of the web in 2018, as teams were growing tired of major version bumps, which happened every three years, requiring full site rebuilds. Prominent agencies began seeing headless as Drupal's bridge to the future – a way to take advantage of Drupal's content modeling and ecosystem capabilities while avoiding costly migrations.

Other major Drupal vendors followed Acquia into headless; annual DrupalCamps set aside a half-day for talks on decoupled approaches.

WordPress: adoption among larger organizations

Like Drupal, by the late 2010s, many WordPress PHP developers were becoming increasingly interested in the world of modern JavaScript. In addition, WordPress was starting to get traction in the enterprise CMS world; headless support helped WordPress integrate smoothly into complex content stacks.

Like WordPress as a whole, headless WordPress has rich functionality – if you know where to look. A community-maintained

Cabrillo College (cabrillo.edu)

Cabrillo College is a northern California community college with 12,000 students across specialities – STEM, business, creative arts, global studies, health and public service.

As the lead web developer, when Ramesh Goonetilleke started Cabrillo's redesign project, he had to think not only about what the site was going to look like, but how it was going to be updated and maintained. Cabrillo has two campuses, technology centers, a library, a bookstore, tutoring, counseling: an arts complex; a dental clinic, a horticulture center, and a disability learning center. Goonetilleke had no desire to play content traffic cop; instead, he wanted to hand the keys of the website section for each part of Cabrillo over to the respective faculty, administrators, and students in charge.

In order to do this, the Cabrillo team created React components building blocks, which they loaded into Gutenberg to give content creators a visual page builder. And for the production version of the site, they pulled these components into a Gatsby site that loaded quickly for the Cabrillo community.

API, WPGraphQL, has gained more support than WordPress's official API. Content modeling requires an additional paid plugin, called Advanced Custom Fields.

In 2020, WordPress's creator, Matt Mullenweg, stirred up controversy by publicly criticizing Jamstack and headless in a tech press article. Later, Mullenweg walked back some of his comments; since then, the trendsetting agencies within the WordPress community have started to publicly talk about the Jamstack and headless projects they've done with larger organizations.

The enterprise CMS dilemma

Other CMS communities had different reactions to the cloud and the modular web. Some smaller CMS communities have embraced it; others changed little.

A few larger CMS vendors took an alternate approach, trying to create larger suites by gobbling up smaller vendors, an approach referred to as "digital experience platforms."

To some extent, this has worked. But these bolted-on systems are rarely well-integrated, and added functionality has come at the cost of increased brittleness. To paraphrase *Hamlet*, there are more things in heaven and Earth than can be bolted on a legacy CMS.

Savvy CMS users have a dilemma. They can fight their CMS to build high-quality sites. Or they can lobby in their organization to migrate off. Many are beginning to slowly, steadily, migrate to the modular web.

Google's mobile performance carrot

Performance specialists weren't the only ones who cared about faster page loads. After informally evangelizing web

Web design evolves towards programmatic depth

Web design in the early 2010s was graphically rich and often resembled magazine spreads: interleaved elements, textured backgrounds, the web page as canvas.

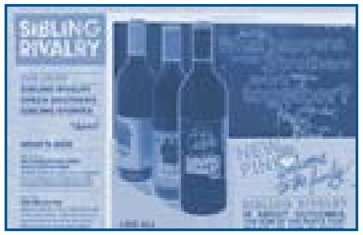
By 2015, responsive design and mobile performance pushed web design away from this style towards flat minimalism. A few elements and a hero image, surrounded by text in a carefully selected font. Instead of representing complexity visually above the fold, it unraveled as the user scrolled down, allowing for more optimised page-loading.

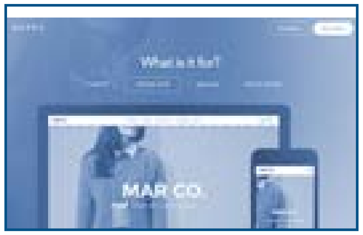
As the decade drew on, designers tried to enhance these lighter weight sites with programmatic depth. 3D illustrations. Lightweight vector drawings, bold typography, emojis, card layouts, drop shadows and flat, vibrant backgrounds.

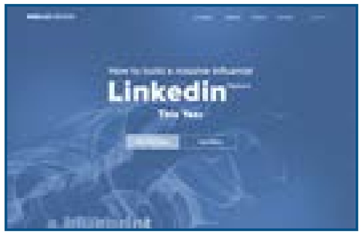
Attention began to be paid to interaction design. CSS micro-animations like border-boxes on hover created an inexpensive sense of depth. Designs emphasized lightness and softness. Visiting them was like walking into a trendy Blue Bottle cafe. Interfaces standardized, with component libraries standardizing elements like CTA buttons, form fields, accordions and dropdowns.

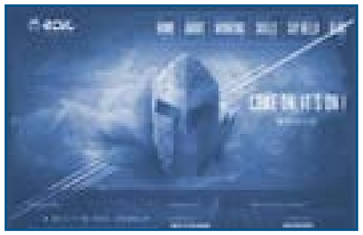
These sorts of stylistic flourishes allowed companies to stand out in a flat, minimalist world – while keeping their sites fast! But using them required buying into the modern JavaScript paradigm. They often relied on JavaScript libraries, complex CSS animations, and component libraries with Storybook to create coherent design patterns.

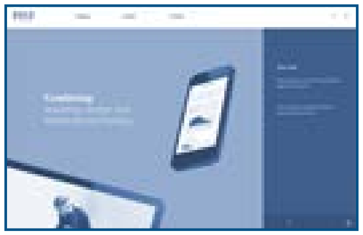
As a result, they were often designed in collaborative web-native apps like Figma, and prototyped with React-based Jamstack frameworks – something with a build pipeline and access to npm. It was just easier to implement effects like progressing image loading, parallax animations, and scrolling transformations in the JavaScript world.

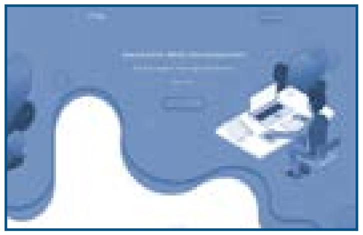












Pre-mobile. 2010

Early mobile-friendly. 2015

Modern mobile-friendly. 2020

performance for over a decade, the 800-pound gorilla of the web started to weigh in. For Google, faster page loads meant fewer bounces, higher ad spend, and more searches.

Around 2015, Google began branding specific performance-related concepts like "progressive web apps." They built testing tools like Lighthouse and PageSpeed insights, and codified metrics, called Core Web Vitals, for measuring page speed. (Google also tried less-effective methods, like their AMP format for news, which became a lightning-rod for criticism and even surfaced in an antitrust lawsuit.)

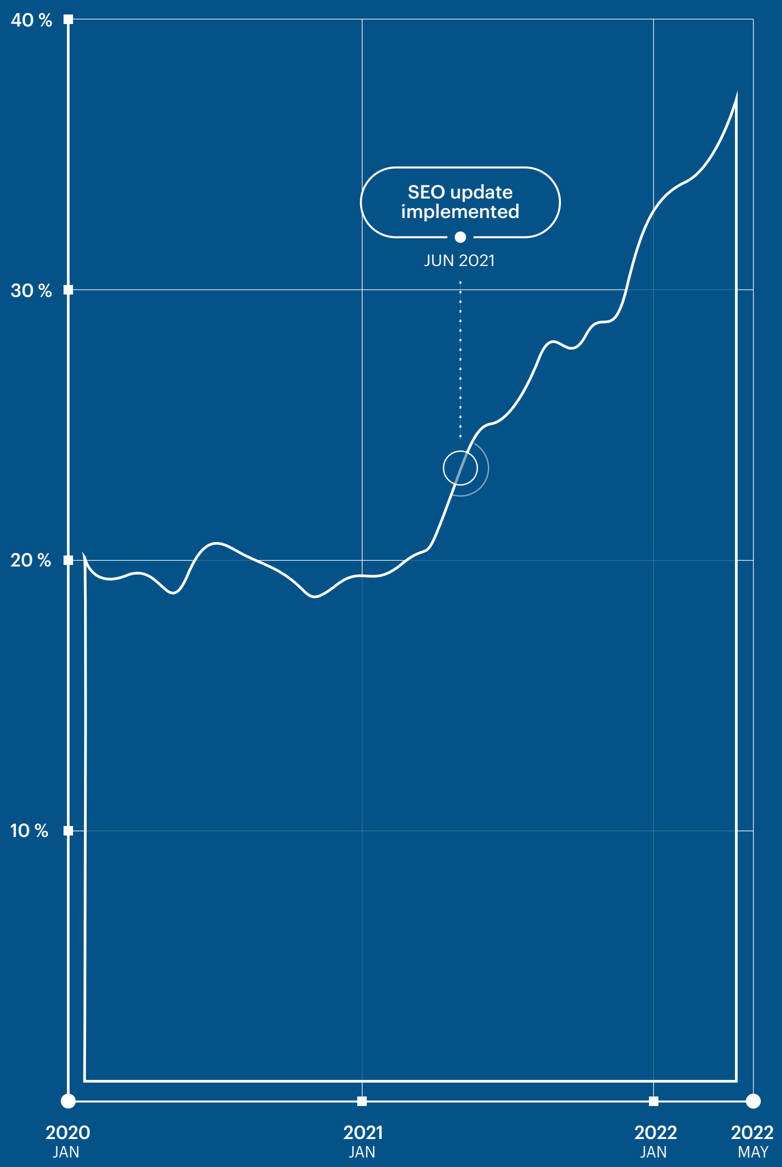
With the web ecosystem hungry for better performance, Google's tools and metrics became defaults – as long as those tools were perceived as platform- and framework- agnostic. Google could specify the "what," but individual developers and publishers wanted to figure out the "how."

In November 2020, Google announced a forthcoming update to the algorithm powering site ranking on search. Known as the Core Web Vitals Search Update, it would reward sites with an average mobile load time below 2.5 seconds, and penalize sites with slower loading times.

The update was scheduled for June 2021; over a 12-month period, the proportion of high-traffic sites meeting Google's performance thresholds almost doubled, from around 20% in March 2021 to 35% by March 2022.

Key takeaway

A number of recent trends are driving adoption of the modular web, including CMSs like Drupal or WordPress with significant community enthusiasm around headless, Google's SEO reward for sites with fast mobile performance, and web design trends pushing toward modern JavaScript.

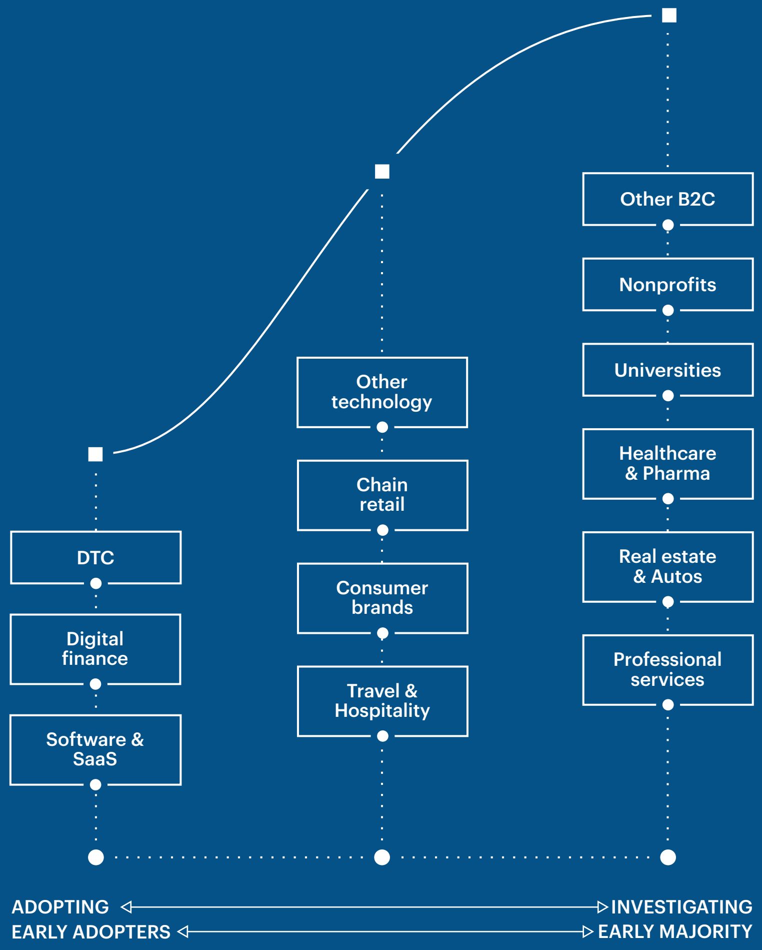


30 % of websites passing Google Core Vitals performance test

CHAPTER 11

MOVING TO THE MODULAR WEB

FINDING YOUR CHAMPIONS • DIFFERENT MODULAR JOURNEYS • INTEGRATING SYSTEMS INTO WORKFLOWS



That brings us to today, to your organization, and your modular web journey.

In the last few years at Gatsby, we've seen thousands of organizations move to this new stack. In this chapter, we'll talk about how their journeys start, and the challenges they encounter – and resolve – along the way.

Some of the topics we'll cover:

→ Different reasons companies turn to the modular web
→ The kinds of personas that drive change
→ How to get a modular setup really humming

There's no one right way – organizations, technology stacks, and team structures vary too much. That said, there are some patterns that we've seen that are fairly universal, which we share in this chapter.

Change starts with a champion

Change starts with one person. It starts to spread when that one person can rally others around the banner. We've seen champions with a variety of motivations and roles. Some used the modular web in previous roles; others just have a sense of what's "next". We've seen a few types of modular web trailblazers:

WEBSITE PROFILE

Hi-Rez Studios

Hi-Rez Studios is a 500-employee game development studio based in Atlanta that creates free-to-play mobile games like Smite and Paladins. Smite has a tight-knit team of three developers responsible for Hi-Rez's entire portfolio of web properties.

In late 2017, the team — Doug, Alissa, and Brandon — tested out Gatsby for a high-profile game launch. The launch went so well that they adopted Gatsby as their default tool, moved over their entire portfolio of highly-visited properties with user-generated content, and unlocked a more interactive experience for enthusiastic gamer communities.

Using a modular web stack let Hi-Rez's website team up their game. With the team able to rapidly deliver whatever the business needed, they began to take on a more prominent role within the company. "About a year in, we started getting a lot of comments about how productive we were," said Smith.

They were entrusted with greater autonomy – able to try new technologies, experiment with different elements of their stack, and invest in staying up to date technically.

"We can always rely on Gatsby and React to be there for us. We're very confident with what we can do now," Alissa tells me. "That's our stack," she adds, pride beaming from her voice.

The Developer Enthusiast. After Joe learned React, he never went back. He led a successful Jamstack technology transition in his last job. A consensus is starting to form that he could *probably* do the same here. It's clear he's technically capable, and well-regarded by the other developers.

Weekend activity	Experimenting with home brewing rig
Their kryptonite	Being stuck in traffic

The Boutique Agency Owner. Christine can usually tell in the first meeting if a potential project would be a good fit. She watches for signs of taste. Attention to detail. Whether potential clients get what makes a great website. Her team wants to be partners, not implementers. Christine wants projects her team can be proud of.

Weekend activity	Something Instagram-worthy, probably with friends
Their kryptonite	Going the extra mile for someone who won't appreciate it

The Architect *(Director of Web Development)*. Rahul knows every in and out of the current website technology stack and development process. He built it, piece by piece, over three years. But today, Rahul has a nagging feeling that he needs to make another change. The site feels dated. The backlog is building up. Junior talent is getting harder to find.

Weekend activity	Taking their kids to a science exhibit
Their kryptonite	Rules that don't make sense

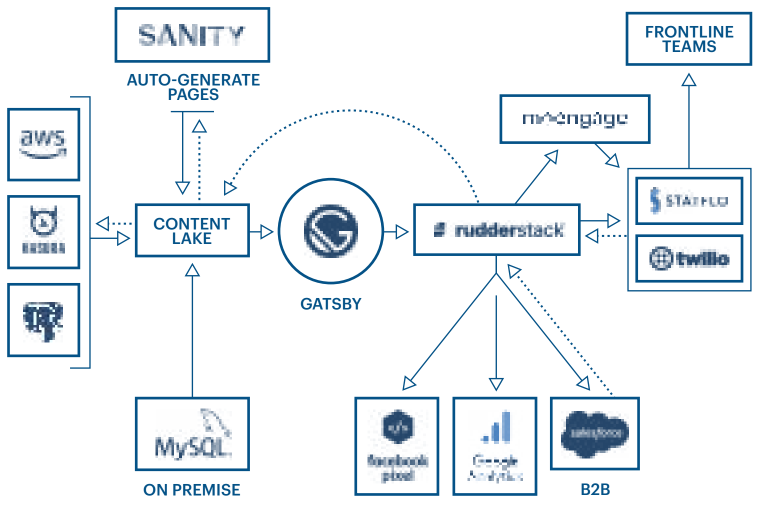
WEBSITE PROFILE

WaveDirect.net

Canadian internet service provider WaveDirect provides high speed internet and TV service to Ontario, including rural communities where they are the only reliable service provider.

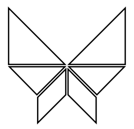
To improve customer communication, the WaveDirect team revamped their website architecture. They designed data and analytics collection, and fed it through Rudderstack, to provide the most important customer data to frontline sales and service agents. They optimized performance with a new Gatsby website. Finally, they drove traffic with SEO content campaigns with their CMS, Sanity.

Metrics at each step of the funnel – website traffic, lead conversion rate, contact response rate – grew by over 50%. Multiplied together, the effect was dramatic – WaveDirect increased new customers from its website by over 4x.

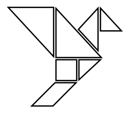


Weaving modular web into the right narrative arc

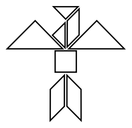
Modular architecture often starts with a one-off project or experiment by an individual or small team. But as modular begins to spread, it can require buy-in from directors, VPs, and benefit from C-suite approval. The key to successful internal advocacy is to frame technical and architectural choices in the narrative of an overall organizational journey. Common modular web journey types include:



The Butterfly: outgrowing a CMS. The organization's veterans agree that five years ago, their CMS worked great. But with the organization launching a slew of digital projects, it seems to be buckling under the load. Every project has brought up critical issues. Two developers just quit in frustration. Everyone agrees that something has got to change.



The Eagle: scanning for growth opportunities. Growth is the organization's buzzword, and almost every commercial team thinks about the LTV-to-CAC ratio. One team experimented with building landing pages in a more modern tech stack. When the new pages converted at 2x the normal rate, the stack they used started spreading...



The Phoenix: refreshing the website, relaunching the company. The company raised a funding round. Hit a revenue milestone. Launched a new product. Brought on a new CMO (or CEO). Changed its target customer. It's goodbye to the old, as the organization reaches for transformation, and puts many of its hopes into a new website.

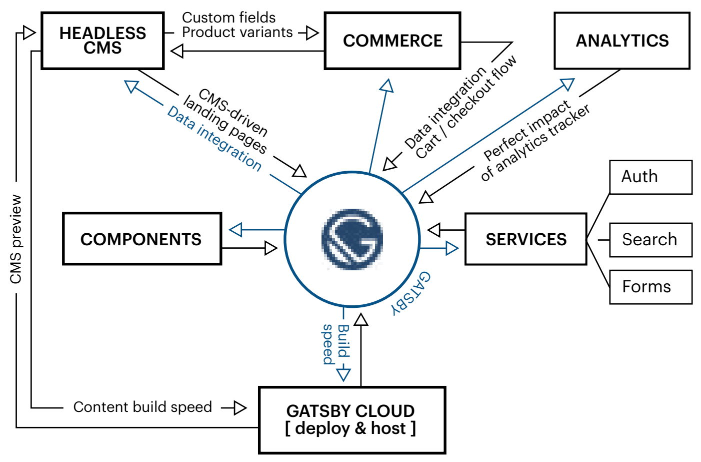
Integrating systems to create seamless workflows

As different functional teams within an organization adopt modular systems, they spend more and more time thinking about the workflows and integrations connecting their systems together.

Headless CMSs begin storing product data, like custom fields or product variant info, that doesn't fit neatly into e-commerce schemas. Developers craft custom flows between headless Jamstack product pages and hosted e-commerce checkout. (29)

Marketers looking to bring content preview and publish times down to seconds start looking toward cloud platforms with **real-time content preview**, deeply integrated with their headless CMS.

Teams create landing page templates in their website framework. They pull in React and Vue components for page sections. Then, they map these elements to content models and fields from their headless CMS, enabling their marketing team to build new pages without writing code.



29 **Relationships between modules in a headless architecture**

To optimize this workflow, they also look for real-time content preview, and turn to framework-specific tricks to optimize performance.

Teams generate different, localized versions of their website, from a single codebase or content space. Or, they use a customized template to generate dozens of similar websites for store locations, faculty members, or new products – known as **"multi-site."** Others generate a **hybrid site** with prominent routes, like their homepage, on modern frameworks, and others on a more traditional CMS architecture.

Build and deploy platforms offered **build-time reports** with diagnostics on site qualities like performance, accessibility, and SEO.

If the webtech architecture is a solar system, these workflow integrations serve as trade routes between planets. They enable interfaces between existing infrastructure and new capacities, share data up and downstream, and help to integrate the whole ecosystem together.

Key takeaways

To drive change, empower champions in and around your organization.

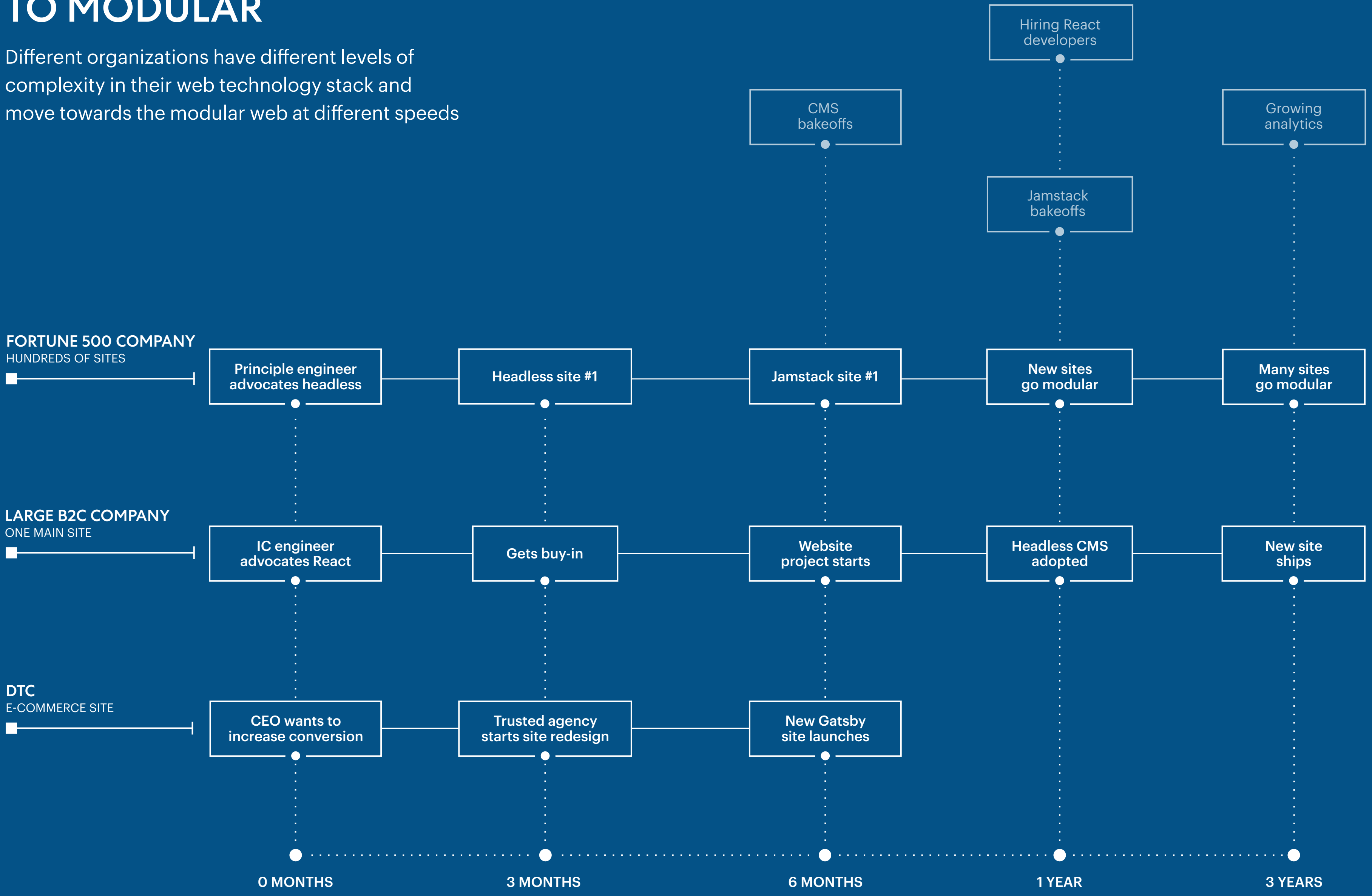
When advocating for modular architecture, center your organization's specific journey type (Butterfly / Eagle / Phoenix).

Different organizations have different adoption speeds, but make sure you are continually making forward progress.

As modular gains momentum within your organization, focus on creating seamless workflows and integrations between systems.

TIMELINE: MOVING TO MODULAR

Different organizations have different levels of complexity in their web technology stack and move towards the modular web at different speeds



QUESTIONNAIRE: ASSESSING YOUR MODULAR WEB READINESS

Every organization's journey is different. However, it likely reduces to a few things.

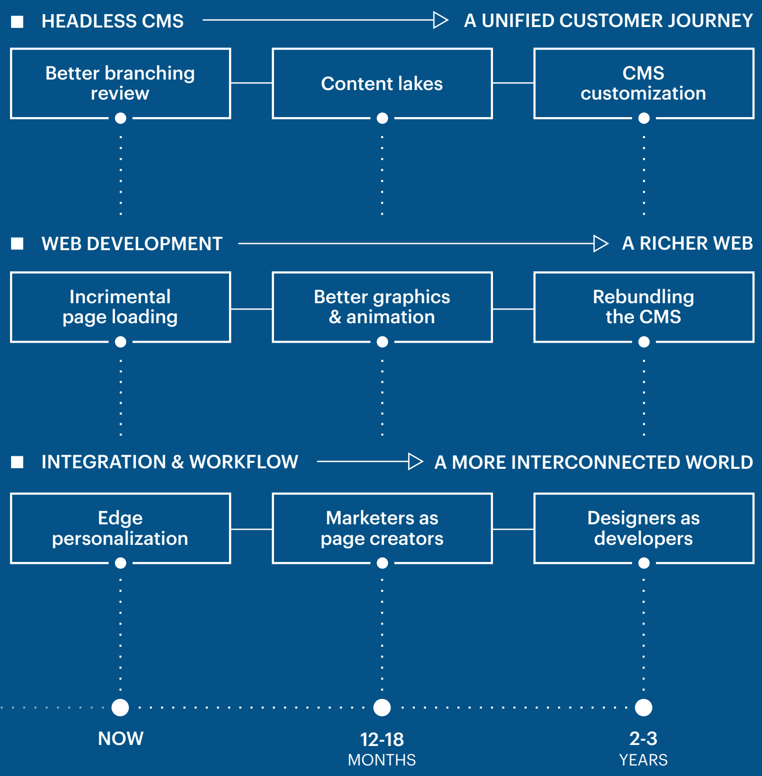
QUESTION	ANSWER	
1. What industry is your company in?	Software, SaaS, startup, DTC e-commerce	10
	Other technology, financial, content/media	8
	Other B2C	6
	University, nonprofit, professional service	4
2. What CMS is your company using?	Headless CMS, or a CMS in headless mode	10
	Drupal	8
	Higher-end CMS (Sitecore, AEM, Optimizely, etc)	6
	Site builder (Squarespace, Wix)	–
	Visual / theme-based page creation within a CMS (eg WordPress themes)	–
3. What is your org's development capabilities?	Strong in-house capability	10
	Trusted agency partner	10
	Confident we can develop capability / agency partner	5
	None	–
4. How familiar is your development team, or your agency's development team, with React or Vue?	Very familiar	10
	Somewhat familiar	8
	Interested to learn	4
	Looking for devs now	4
	Not familiar, not interested	–
5. How many modular web projects have you shipped?	3+	10
	1-2	6
	0	–
6. What are your top 3 priorities for your website right now? (add up score)	Better performance	4
	Faster development cycle	4
	Better lead conversion	4
	Compelling design	4
	Structured reusable content	3
	Better SEO	3
	Improved e-commerce	*2*
	Analytics or tracking	*2*
	Personalization	*2*
	Low-code page creation for marketers	*0*

SCORE	RECOMMENDATION
40+	**Modular-web growing.** Your team has shipped significant projects and is seeing the business value of the modular web.
30-39	**Modular-web ready.** You have the desire, organizational capability to deploy the modular web in your organization – you're probably already planning to use the modular web on major projects coming your way.
20-29	**Modular web knowledgeable.** You've familiar with the modular web and likely already planning initial projects or proof-of-concepts to deepen your experience.
10-19	**Modular-web curious.** You're interested in the modular web, but may need to level up your organizational capability by seeking out a trusted agency partner or finding talent with Jamstack experience.

CHAPTER 12

THE WEB'S NEXT PARADIGM

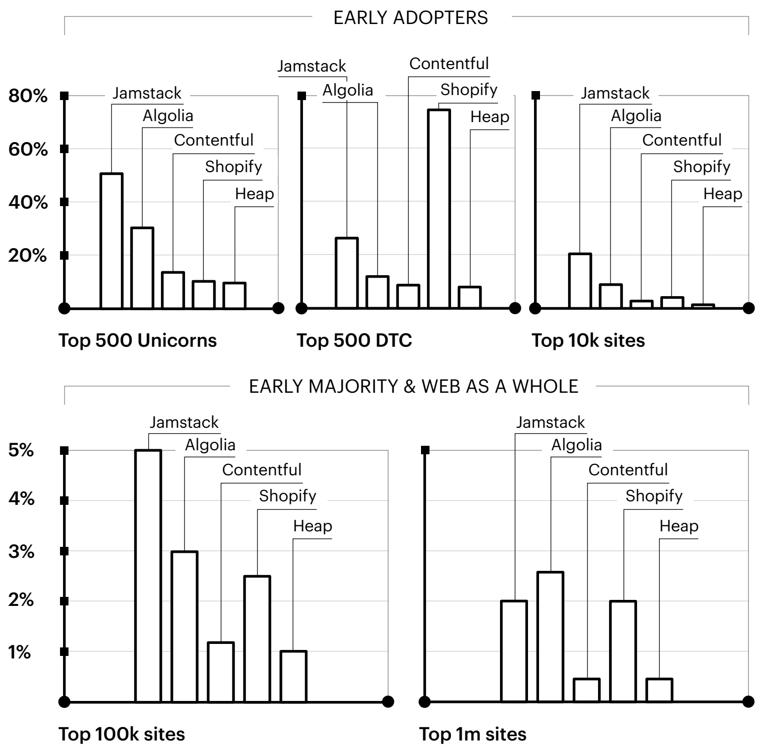
WHERE WE ARE NOW • WHAT'S NEXT • WHERE WE'RE GOING



Where We Are Now

The modular architecture has become the default among early adopters, with leading vendors having attained huge market share among technology startups and DTC e-commerce.

But it's also quickly spreading to the web at large – first to the more results- and social-proof oriented early majority: financial services, online media, travel, retail, donation-funded nonprofits, household consumer brands. (30)



30 Modular web usage

How did we get here?

An analogy is useful. Today's consumer drones were enabled by the early-2010s smartphone boom making hardware components plentiful and cheap. So too, modern websites are enabled by the early-2010s cloud/SaaS boom in modular web technology.

E-commerce, headless CMS, modern JavaScript, build and deploy platforms, marketing analytics – these are all ecosystems now, full of agencies and conferences, collaborators and competitors, everyone riffing together. (31)

With vibrancy comes progress. Today the modular web is replacing – not just supplementing – the CMS.

What's Next

Web development: more performance, more interactivity

The performance edge of the modern web over the CMS stack is wide. It will continue to grow. Three trends in particular stand out.

Incremental page loading is putting content onto mobile screens faster. Smoother "progressive" image loading took off in 2020; now, frameworks are developing tools to prioritize rendering different parts of a page. Text and images might load first, then auxiliary elements (like social share icons), with marketing scripts delayed until *after* the page is interactive.

3D graphics and animation. Graphics and animation cost computing cycles. Flash died in the early 2010s because it was slow on mobile. The web animation world is starting to recreate its functionality – a decade of Moore's Law later.

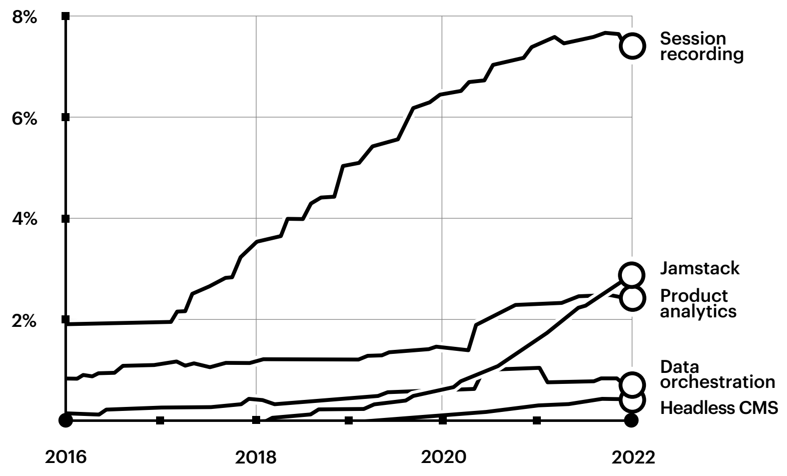
Edge compute and orchestration. In search of faster

performance, apps are moving from the data center to the edge, from web servers to serverless functions orchestrating third-party services, from cloud databases to serverless databases and app caches. The architecture can be complex – but the speed gains are unambiguous.

The headless CMS: maturing pipelines and workflows

Modern physics learned that light exhibits particle/wave duality – both identities can be used to describe its behavior. Similarly, content is both *data* and *code*. It can be piped between systems, and collaborated on by teams. The dual nature of content will drive many of the trends over the next few years.

Content aggregation. Like a "data lake" architecture, centered around Snowflake or Google BigQuery, we'll begin to



(31) **Propuct categories by adoption**
Data from BuiltWith, aggregated across 30 technologies

TECH FEATURE

Tools Worth Tracking

	TREND / WORKFLOW	PROMISING TOOL / FEATURE
WEB DEVELOPMENT	Incremental page loading	↘ Gatsby & Next.js Scripts ↘ React Server Components
	3D graphics & animation	↘ Lottie ↘ Rive
	Edge compute	↘ Gatsby ↘ Next.js Functions ↘ Edge Middleware (multiple vendors)
HEADLESS CMS	Content lakes	↘ Sanity Content Lake ↘ Gatsby Cloud "Valhalla"

	Content branching & review	→ Releases & Workflows
		  ↘ Contentful
		  ↘ Contentstack
		  ↘ Sanity

	CMS customization	→ Apps
		  ↘ Contentful
		  ↘ Contentstack
		  ↘ Dato
		→ Sanity Plugins & Portable Text

INTEGRATIONS	Designer-developer handoff	↘ Figma React Export ↘ Storybook
	Marketer drag & drop UIs	↘ Builder.io ↘ Gatsby Cloud Preview ↘ Prismic Slice Machine
	Edge personalization	↘ Uniform ↘ Ninetailed

see larger organizations turning to a "content lake" aggregating content and data from all of their systems. This will be a boon to organizations with diverse content sources, or who rely on mission-critical systems with tricky APIs (homegrown services, many e-commerce platforms).

Better branching and review. Headless CMSs started with two states for content: Draft and Published. They are moving toward richer review and approval processes, more customizable environments, and support for releases of multiple pieces of content at once.

CMS customization and extension. Mature ecosystems like Salesforce Exchange and Shopify's App Store, feature *thousands* of apps and integrations, including vertical-specific tools and workflows, analytics integrations, deep layout customizations, and page builder UIs. Headless CMS ecosystems are much younger, with dozens of integrations, but will develop this functionality over time.

Better integrations & workflows

The biggest challenge facing modular web users today is the complexity of managing all the moving parts. The last 2-3 years have brought new integrations and workflows (see Chapter 11). The next few years will bring more.

Prototypes that designers can give to developers. Designers working in pixel-based tools like Figma "hand off" drawings to developers, who recreate them with code. While freelancers (who do both design and development) have been exporting components from design tools to React/Vue for years, workflows haven't yet solidified for larger organizations with separate, dedicated roles.

Easier-to-set-up drag-and-drop experience for marketers. Most website teams are happy with their landing page builder

– once they create a page template. But setup can be complex, requiring good engineering judgment and wiring multiple different systems together. Luckily, services simplifying the process are starting to emerge; over time, this functionality will likely grow into the app ecosystems of headless CMSs and build & deploy platforms.

Edge personalization. A previous generation of "client-side" A/B testing tools like Optimizely, which delayed page loading in order to run experiments, is being replaced with personalization built on edge-capable serverless platforms.

Where We're Going

As the modular web spreads, it has the potential to fundamentally change some assumptions we make: about business, technology, and the web.

Unifying your customer journey

In 1957, Walt Disney sketched out one of his most important illustrations. It wasn't the basis for a new cartoon – it was a diagram of the Disney user experience, with boxes for the Disneyland park, audiovisual experiences like films and TV series, and other branches like merchandise, connected with arrows.(32)

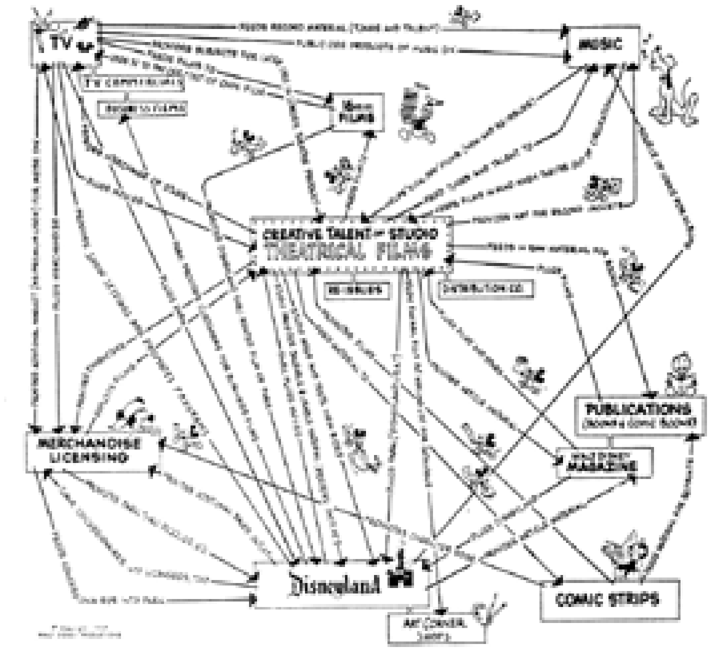
Walt Disney's intuition led him – decades ahead of the rest of the world – to a key realization about business success. He realized that you have to create great experiences – and then tie those experiences seamlessly together.

In the Disney universe, completing one experience (watching *Frozen*) makes it more likely for you to complete another experience (buying Anna or Elsa dolls, watching *Moana*, visiting Disneyland). The underlying assets of one experience (character

animations) make it easier to create another experience (a sequel).

The modular web is leading companies along the path Disney saw decades ago. It's moving companies to a world where website building blocks – page templates, components, content, commerce – are modular, reusable, centrally organized, discoverable, and interoperable.

In this world, companies launch new digital projects in weeks rather than months. Teams can create experiences around single products, or connect customer journeys through an entire product line or catalog.



32 **Walt Disney's 1957 sketch of the Disney user journey**

In short, the modular web provides a set of tools for understanding, creating, enriching, and connecting customer experiences. By bringing designers, developers, content, and marketing teams together, it tightens the feedback loop between what customers want and what businesses do. **The modular web helps companies unify their customer journey.**

A richer and more powerful web

The web is the world's largest open technology platform. In the mid-2000s, "Web 2.0" promised that blogs and social networks would empower Internet users to connect and collaborate.

Today, many believe that Web 2.0 has backfired. Tech giants like Google and Facebook seem all-powerful relative to ordinary users, developers and businesses.

More controversially, advocates of "Web 3.0" believe that a better web will be made possible by programmable cryptocurrencies, five or ten years in the future. Whether or not this pans out, it's clear that the modular web is a sort of "Web 2.5:" a huge step forward for web technology that is *already* widely used with significant benefits: more open, more dynamic, more decentralized.

A truly global web. High-performance Jamstack sites hosted on global CDNs are usable not just in New York or Paris, but in Jakarta, Lagos, and Mexico City, to people using a budget Android device, with 3G and limited bandwidth.

Richer and more interactive. Component-based UIs and web services are adding more interaction, better search, more accessibility, more personalization. Better performance is enabling more "multiplayer" experiences, richer media, and more immersive experiences.

A gateway to cloud computing. Twenty years ago, the LAMPStack democratized websites and applications. But today, most high-end, dynamic, interactive experiences are – again – being built by big technology companies. Today, Jamstack and headless CMSs allow individuals and businesses to craft unique, globally scalable experiences.

Toward a more interconnected world, together

In 2009, when Shopify launched their app store, they were a tiny vendor with ten employees. Today, Shopify is the most widely used e-commerce system in the world.

In 2013, when Contentful launched, the cloud model and API-driven companies were in their infancy. Today, the headless CMS is the clear successor to the monolithic CMS.

When Gatsby.js was first published on Github seven years ago, React was a marginal JavaScript library. Today, React is the most popular UI framework on the planet.

Five years ago, when Gatsby launched as a company, the Jamstack was seen as an odd developer-centric architecture. Today, it's the clear choice for fast, scalable sites.

Change can come slowly – and, then, all at once. But the modular architecture is here to stay. And we're here to help, especially with making it all work together.

For us, the first piece was integrating Gatsby with headless CMSs. We launched our plugin system with CMS integrations, coined the term "content mesh," and launched content preview. Then, we focused on builds and hosting, getting new code live in minutes and new content in seconds.

Soon, we'll be exposing our data layer as an API, so that any content or data being pulled into a Gatsby project can also easily be used for non-Gatsby – even non-web – projects. We're working on better script and performance optimization, and integrating other categories of tools like A/B testing and personalization.

And if you're trying to figure out where to start, or how to proceed – anywhere in this space – you can reach me at sam@gatsbyjs.com. But it's not just me, or even Gatsby. It's a whole ecosystem of freelancers, agencies, and technology vendors. A diverse group of happy users. Industry alliances like MACH.

The web's last decade has brought innovation. New web building blocks. E-commerce. Headless CMS. JavaScript frameworks. The Jamstack. Global CDNs. Search, auth, and other web services.

The web's next decade will bring transformation. This new architecture will give rise to hundreds of thousands, or millions, of new, unique web experiences. Together, we can move away from walled gardens to create a more diverse, interconnected digital world.

ACKNOWLEDGEMENTS

This book would not have made it to press without my editor, Gary Zhezi Zhang, and my illustrator, Vlad Afasaniev. Trying to write a book while also managing a number of other duties has been difficult, but Gary and Vlad's high-quality, rapid feedback made it possible.

It's hard to thank everyone in the modular web ecosystem, but I'd like to thank my co-founder Kyle Mathews for inviting me on this wild ride, and each of my colleagues for coming along and working together to build the future of the web. Thanks in particular to those whose feedback on the text was invaluable – Zack Urlocker, Dustin Schau, Maddie Wolf.

This book is a sequel of sorts to two different books: Deane Barker's *Web Content Management*, the canonical guide to the CMS of the 2010s, and Brian McCullough's *How the Internet Happened*, a history from Mosaic to the iPhone launch. Another inspiration was Nadia Eghbal's *Working in Public: The Making and Maintenance of Open Source*.

One of the best parts of both my role and writing this book is the ongoing discussion about the past, present, and future of the modular web. In particular, I want to thank a number of people whose insights informed the text: Sascha Konietze, Bridget Perry and Nicole France at Contentful; Jason Cohen at WPEngine; Matt Baier and Nick Barron at Contentstack; Nick Genert at WPVIP, Dries Buytaert of Acquia; Josh Koenig of Pantheon; Knut Melvaer at Sanity; Nouha Chhih and Angelo Ashmore at Prismic; Victoria Duggan at Shopify; Mark Demeny at Uniform; Eric Dodds at Rudderstack; Shawn Wang formerly of Netlify; Addy Osmani at Google; Kaya Ismail at CMSWire, Tony Byrne at Real Story Group, Robin Vasan at Mango Capital; Justin Emond at Third And Grove, Brian Webster at Delicious Simplicity; Thomas Slade at Elevar, Kevin Gervais at Touchless, and Trevor Health of Novvum.

Writing a book is, if you can structure it properly, a task of O(n log n) complexity. If you can't, it's probably O(n^2) and you'll never finish. For avoiding that pit I owe thanks to my most thorough editors – Tim Crews (RIP) at the *Sacramento Valley Mirror* over a decade ago and Venkatesh Rao at Ribbonfarm only slightly more recently – and my mother, Haika Gay, who first taught me to embrace red ink.

Lastly, to the hundreds of thousands of developers, designers, and marketers who are building modular websites professionally, or working for modular web vendors. Your time and your talents are building the future. Thank you.

ENDNOTES

CHAPTER 2

Heterosexual dating data from Stanford research study
the study is “Disintermediating your friends: How Online Dating in the United States displaces other ways of meeting” in PNAS, August 2019, by Michael J. Rosenfeld, Reuben J. Thomas, and Sonia Hausen.
↗ https://doi.org/10.1073/pnas.1908630116.
They asked people in two waves: one group in 2009 and another group in 2017. (or more; maybe some “bar/restaurant” couples just don’t want to admit it).

Yishan Wong’s statement
is from an answer he wrote on Quora in 2010:
↗ https://www.quora.com/Without-mentioning-coupons-why-and-how-is-mobile-the-future-Or-will-the-next-revolution-really-be-based-around-deals-and-monetary-incentives
Known in the industry as “Yishan”, Wong is one of Silicon Valley’s more interesting characters. He was a member of the so-called “PayPal Mafia”; then one of Facebook’s first 50 employees; then the CEO of reddit; today, he runs a climate change startup.

Mobile visits as a % of total site traffic
is from StatCounter
↗ https://gs.statcounter.com/platform-market-share/desktop-mobile-tablet/worldwide/2011

Daily time on device
is from a study/forecast by Zenith in 2019, available on Statista
↗ https://www.statista.com/statistics/319732/daily-time-spent-online-device/

CHAPTER 3

CMS taxonomy
from Deane Barker, “Web Content Management: Systems, Features, and Best Practices” (O’Reilly, 2014). Barker’s book – written as the CMS was rising and ascendant – is the definitive guide to managing web content in the 2010s.

Share of web
from w3techs’ historical yearly trends data:
↗ https://w3techs.com/technologies/history_overview/content_management/all/y

CHAPTER 4

SaaS funding data
from Christoph Janz: “Trying to make sense of 10 years worth of SaaS funding data”, Medium, Jan 14, 2018
↗ https://medium.com/point-nine-news/trying-to-make-sense-of-10-years-worth-of-saas-funding-data-in-14-simple-charts-c509de1d6386
Janz is respected as a leader in European early-stage venture capital; he led the first round of funding into Contentful.

Contentful
July 2011 Interview (in German) with Konietzke:
↗ https://www.foerderland.de/gruendung/news-gruenderszene/viele-agenturen-haben-uns-bereits-anfragen-fuer-eine-white-label-loesung-von-storageroom-gestellt/;
Konietzke’s June 2013 launch post:
“From StorageRoom to Contentful”
↗ https://www.contentful.com/blog/2013/06/23/storageroom-to-contentful/; personal conversation.

Stripe
Patrick Collison tells Stripe’s founding story to Derek Anderson of Startup Grind in a 2012 interview:
↗ https://www.youtube.com/watch?v=Ubo9UaRxdVg

CHAPTER 5

What saved Facebook’s stock price…
quote and information are from Garcia-Martinez’s memoir Chaos Monkeys, as is the subsequent tidbit about Fidji Simo.

Facebook mobile vs desktop earnings numbers
are from quarterly call transcripts on Seeking Alpha.

Along the way, Dunn started a conversation...
Dunn's May 2016 Medium article "The Book of DNVB"
is recognized as the category creating piece.
↗ https://dunn.medium.com/digitally-native-vertical-brands-b26a26f2cf83

The New York Times wrote in 2017...
from "How Facebook's Oracular Algorithm Determines the Fates of Start-Ups", Burt Helm, Nov. 2, 2017,
↗ https://www.nytimes.com/2017/11/02/magazine/how-facebooks-oracular-algorithm-determines-the-fates-of-start-ups.html

Mobile commerce having grown
by roughly 10x over five years
is from comScore's 2020 State of Retail Preview,
available on Statista:
↗ https://www.statista.com/statistics/268439/quarterly-us-mobile-e-commerce-spending/

...still only represented 15% of online purchase dollars...
Andrew Meola, Oct 2016, Business Insider:
↗ https://web.archive.org/web/20161117083611/https://www.businessinsider.com/mobile-commerce-shopping-trends-stats-2016-10

Before the cloud
The best summaries here were from investment documents: Cloudflare's Aug 2019 S-1
↗ https://www.sec.gov/Archives/edgar/data/1477333/000119312519222176/d735023ds1.htm
and Fastly's Apr 2019 S-1
↗ https://www.sec.gov/Archives/edgar/data/1517413/000119312519111675/d702138ds1.htm
Usage data from w3techs reverse proxy
↗ https://w3techs.com/technologies/overview/proxy

Average mobile connection speed
from Nielsen Norman:
↗ https://www.nngroup.com/articles/the-need-for-speed/
median mobile page weight and average mobile load time from

HTTPArchive data:
↗ https://httparchive.org/reports/page-weight?start=earliest&end=latest&view=list#bytesTotal
↗ https://httparchive.org/reports/loading-speed?start=earliest&end=latest&view=list#ol

CHAPTER 6

But JavaScript was changing quickly
May 2020 blog post by Shawn Wang,
known more commonly as swyx
↗ https://www.swyx.io/js-third-age/

In a 2011 meetup talk
Brendan Eich, Sep 2011 CapitolJS, presentation at
↗ https://www.slideshare.net/BrendanEich/capitol-js.

A StackOverflow analysis...
from "The Brutal Lifecycle of JavaScript Frameworks",
Ian Allen,
↗ https://stackoverflow.blog/2018/01/11/brutal-lifecycle-javascript-frameworks/

Language chart
from StackOverflow Trends:
↗ https://insights.stackoverflow.com/trends?tags=javascript%2Cruby-on-rails%2Cphp%2Casp.net.
Framework chart also from StackOverflow Trends:
↗ https://insights.stackoverflow.com/trends?tags=angular%2Cvue.js%2Creactjs%2Cangularjs

CHAPTER 7

Most of this chapter is from personal experience and conversations with Kyle Mathews.

Housecall Pro
from "Growing Housecall Pro by 973% with Gatsby" by Cory Mortimer on the Gatsby blog:
↗ https://www.gatsbyjs.com/blog/2019-05-02-growing-housecall-pro-by-973-percent/

Jaxxon
from a conversation with project lead Trevor Heath, published on the

Gatsby blog:
↗ https://www.gatsbyjs.com/blog/jaxxon-gatsby-shopify-faster-growth/

CHAPTER 8

In a 2016 blog post on headless...
"Use Cases for a Headless CMS"
↗ https://deanebarker.net/tech/blog/use-cases-for-headless-cms/

Most marketing teams start with an analytics use case
is from "The tools of today aren't the tools of tomorrow" by Calvin French-Owen on the Segment blog:
↗ https://segment.com/blog/the-tools-of-today-arent-the-tools-of-tomorrow/

The webtech solar system
is from a personal conversation with Bridget Perry, Feb 6, 2022.

CHAPTER 9

Shopify merchants per year
pulled from Shopify quarterly results, available at MarketplacePulse:
↗ https://www.marketplacepulse.com/stats/shopify/shopify-number-of-merchants-41

Shopify as % of the web
from w3techs:
↗ https://w3techs.com/technologies/history_overview/content_management/all/y

Nike started selling online
stats from Statista:
↗ https://www.statista.com/forecasts/1218320/nike-revenue-development-ecommercedb

Tempur Sealy
stats from quarterly earnings calls and reports.

Strivectin.com
from Thomas Slade's post on the Elevar blog: "Headless Shopify: Lessons Learned Building with Gatsby, Part 2"
↗ https://www.getelevar.com/shopify/headless-shopify-learning-lessons/
and Garrett Dimon's blog post on the Gatsby blog: "How Elevar Prototyped a Headless E-Commerce Store with Gatsby"

↗ https://www.gatsbyjs.com/blog/how-elevar-prototyped-a-headless-e-commerce-store-with-gatsby/

Revenue from online channels
from Target, Walmart, P&G etc from quarterly earnings transcripts, accessed via Seeking Alpha.

Little Caesars story
from Michelle Gienow's blog post on the Gatsby blog, "Little Caesars Pizza Delivers with Gatsby,"
↗ https://www.gatsbyjs.com/blog/2020-07-15-little-caesars-delivers-with-gatsby/

CHAPTER 10

Taking a site from one major Drupal release to another...
from Josh Koenig's Jan 2013 post "Drupal 8: Upgrade Planning and Best Practices", on the Pantheon blog:
↗ https://web.archive.org/web/20211007231029/https://pantheon.io/blog/drupal-8-upgrade-planning-and-best-practices

Drupal project founder and Acquia CTO....
from "The future of decoupled Drupal", Dries Buytaert, September 24, 2015,
↗ https://dri.es/the-future-of-decoupled-drupal.

In a follow-up blog post...
"How to decouple Drupal in 2019", Dries Buytaert, January 2019,
↗ https://dri.es/how-to-decouple-drupal-in-2019

In the first week of 2021...
"Is Graphic Design the C.I.A.'s Passion?" from Ezra Marcus in the New York Times, Jan 2021:
↗ https://www.nytimes.com/2021/01/08/style/cia-rebrand.html

Homepages from popular "Top Web Design Trends" lists throughout the 2010s...
"The State of Web Design Trends: 2011 Annual Edition", Brandon Jones, Dec 30, 2010,
↗ https://webdesign.tutsplus.com/articles/the-state-of-web-design-trends-2011-annual-edition--webdesign-1610;
"The 10 Big Web Design Trends of 2015", Jerry Cao,
↗ https://www.sitepoint.com/the-10-big-web-design-trends-of-2015/;

"10 innovative web design trends for 2019", Lennart de Ridder,
↗ https://99designs.com/blog/trends/web-design-trends-2019/

In November 2020, Google announced a forthcoming update...
↗ "Timing for bringing page experience to Google Search", https://developers.google.com/search/blog/2020/11/timing-for-page-experience

The proportion of high-traffic sites meeting Google's performance thresholds almost doubled...
from HttpArchive Core Web Vitals Technology Report,
↗ https://datastudio.google.com/s/tEN69TiwhkQ

CHAPTER 11

Hi-Rez Studios
from conversations with Smith, Perry, and Perry, in
↗ https://www.gatsbyjs.com/blog/case-study-hi-rez-studios/

Over time, as teams with an organization began adopting systems...
from author's blog post on the Contentful blog: "Ship your Contentful website faster with incremental architecture",
↗ https://www.contentful.com/blog/2022/06/07/ship-your-website-faster-with-incremental-architecture/

Canadian internet service provider WaveDirect...
from author's blog post on the Gatsby blog:
↗ https://www.gatsbyjs.com/blog/how-wavedirect-used-gatsby-rudderstack-and-sanity-to-4x-leads-and-dominate-search-results/

CHAPTER 12

While the modular architecture is concentrated among early-adopters...
unicorn startup list pulled from Crunchbase unicorn list; DTC list from 2PM DTC List; accessed Jun 18, 2022; usage data from BuiltWith.

Product categories by adoption...
usage data from BuiltWith. Author wrote a script that scraped graphs BuiltWith raw HTML and converted it to the underlying site data. Products were grouped by BuildWith categories.

SAM BHAGWAT

MODULAR
The Web’s New Architecture
And How It’s Changing Online Business

Edited by Gary Zhexi Zhang
Cover image and design by Vlad Afanasiev