the no-code revolution

webflow

webflow

Break the code barrier — webflow.com

Need help or have a question?

We're here to help on our forum, University, and at contact@webflow.com.

Contents

Special thanks to these individuals for making this book a reality:

Jeff Cardello
Camille Esposito
Shannon Fisher
Omid Ghiam
Johnnie Gómez
David Head
Barrett Johnson
Meg Wehrlen
John Moore Williams
Ryan Morrison
Saree Rice
Cameron Roe

CHAPTER I

Foreword

Ironically enough, although it gets talked about a lot on Twitter and feels like “the hot new thing,” the “no-code movement” is not all that new.

— Barrett Johnson, Product Marketing Manager

At its core, the no-code movement is just an evolution of core principles that has driven technological innovation for millennia: namely, the desire to democratize and scale processes, tools, and access to mediums which were formerly available only to a small set of people — and through that democratization, multiply the potential of what humankind can create.

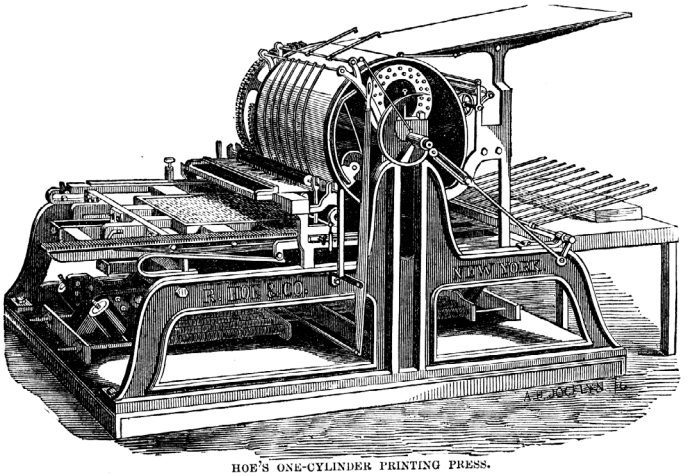
Democratizing mediums to unleash creative potential

Consider the following examples of tools and technologies that we've created to democratize access to formerly inaccessible mediums. Spoiler alert: **the more access people have to a medium, and the easier we make that access, creativity and output and innovation go through the roof.**

The democratization of publishing

Before the advent of the printing press, the only way to mass produce books was by hand. This made the spread of new

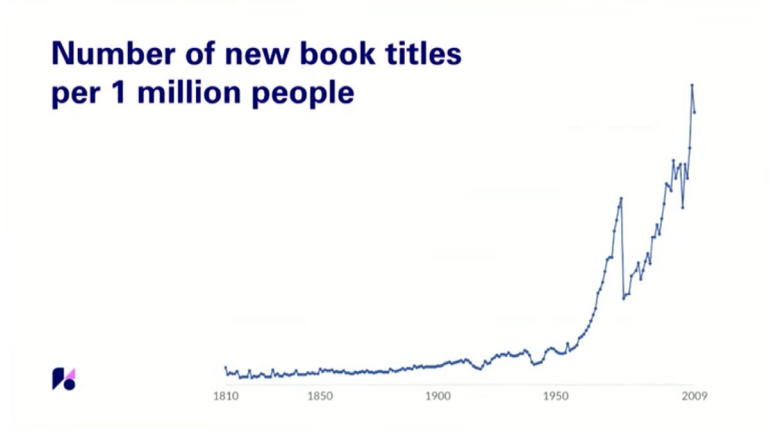
information and knowledge not only slow, but also controlled by the institutional powers of the day (in the case of Europe, this was the church).



HOE'S ONE-CYLINDER PRINTING PRESS.

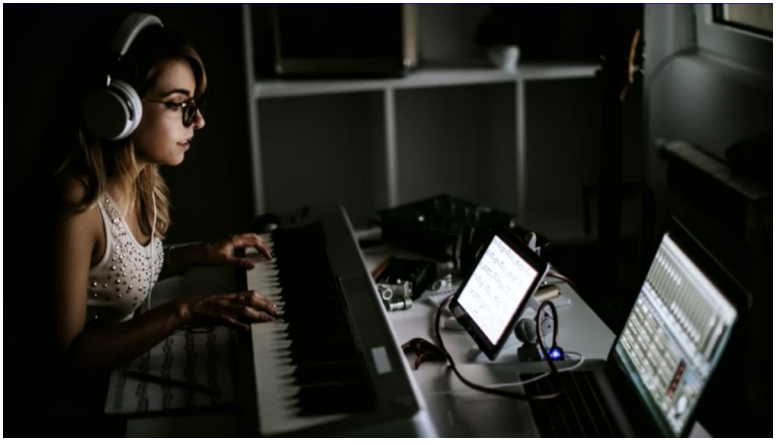
The invention of the printing press in 1440 made the mass production and distribution of new texts possible in a dramatic new way. And if you were to trace the thread of publishing to the modern world, through typewriters and now word processing technology, publishing your ideas to the world is only a few clicks away.

The results of this increasingly accessible world of publishing are unsurprising: consider the rise in the number of new book titles per 1 million people in the last 200 years.

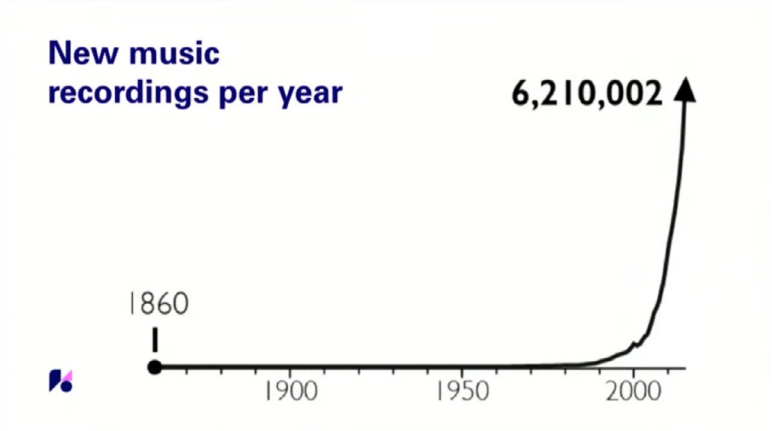


The democratization of music production

Similarly, in the realm of music, technology and access have had a dramatic impact on the volume of music that we create. In the past, creating a record would have required industry connections, significant capital for studio time, and more.



Now, with the advent of home recording technology and publishing platforms like SoundCloud and Spotify, creating and sharing music is easier than ever.

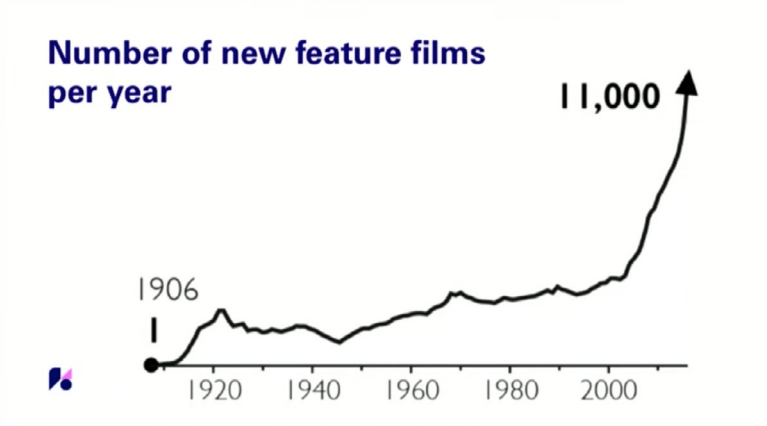


The democratization of movie production

In the realm of movie production, the pattern has been the same. In the past, creating movies was an extraordinarily expensive affair, and so production was limited only to those who could afford it or fundraise significant amounts of capital.

As cameras have become more available, and independent filmmakers have had a chance to share their ideas without so much upfront capital, creativity has blossomed.

Nowadays many people have a video camera in their phone — and the rise of movie production in recent history attests to the fact that greater access to a medium impacts the output and creativity of an entire industry.



The democratization of software

The no-code movement continues this evolution — but simply in the realm of software. **Put simply, the no-code movement rests upon the fundamental belief that technology should enable and facilitate creation, not be a barrier to entry.** The potential for this movement, judging by the examples laid out above, is tremendous — and for those of us participating in shaping it, we really do have the potential to change the world.

CHAPTER II

What is the no-code movement?

And what it means for modern businesses and marketing teams.

— John Moore Williams, Senior UX content strategist

> *These tools are reducing the amount of time and coding expertise required to translate an idea into something people can use. You no longer need to become a programmer to build things on the internet, empowering a new wave of makers from different backgrounds and perspectives.*
>
> — Ryan Hoover, founder of *Product Hunt, in The Rise of "No Code"*

In the early 1970s, Steve Jobs and Steve Wozniak exhibited the first Apple II at the First West Coast Computer Faire in San Francisco. The Apple II boasted built-in BASIC programming language, color graphics, and a 4100-character memory for *just* $1298. Programs and data could be stored on an audio-cassette recorder (remember those?!). Before the end of the fair, Wozniak and Jobs had secured 300 orders for the Apple II and from there Apple just took off.

Tandy Radio Shack and IBM weren't far behind, quickly producing their own consumer-targeted computers.

What united all three machines, though, was their complete lack of a visual interface. Everything happened in the equivalent of the command line. That limited the new home computer's utility to the technically savvy: people capable of using BASIC to write and execute their own home-rolled programs and share them with others.

It wasn't until 1984 that the true potential of the computer was unleashed, again by Apple, with the first iteration of the Macintosh.

What set it apart? The graphical user interface (or GUI, for short). And a little thing we now call the mouse.

Enter the visual interface

It's probably wildly obvious *why* the visual interface (or GUI) changed *everything*, but just to make sure we're all on the same page:

It democratized the computer by lowering the knowledge barrier between the average person and daily use. Instead of having to code your own programs — or use those others shared, but still had to know how to install — you could suddenly just fire up your machine and start clicking. It didn't completely remove the barrier to entry, of course. Digital literacy courses continue to run in schools all over the world, and they're becoming increasingly valuable in the era of "fake news."

But it's also *radically* easier to learn and use a computer when it's all visual. You can click around and *see* what happens.

In what amounts to a blink in a historical timeframe, the computer went from a specialist tool usable by only a fraction

of the world to something literally anyone with vision and a modicum of manual dexterity could employ regularly. (This is not to downplay the significance of those remaining barriers to entry. They're wildly important and something we're still working on addressing properly.)

If you're at all familiar with Webflow, this might start to sound familiar. We're essentially trying to move a text-based interface model to a visual (GUI) one.

What does this have to do with the "no-code" movement?

The brief history lesson above was included to contextualize the no-code movement, which essentially consists of a series of emergent tools that aim to transform typically code-based workflows and empower anyone to handle them, without writing code. Hence the name.

If you ask many people in the worlds of design, startups, and tech, you'll encounter some who are discomfited by the movement. That's natural. Change is hard.

But from a historical viewpoint, it's just the latest iteration of what increasingly seems like a natural arc in the development of any technology. What begins with highly specialized toolkits

and knowledge bases is gradually expanded, simplified, and designed to be simpler and more accessible.

Just look at the history of printing and its business outgrowth, publishing. What began in pre-common-era China hit Europe in the late middle ages and radically transformed culture, making the transmission of knowledge dramatically easier than the manuscript model. But it still required expensive equipment and knowledge that required years of training to properly apply. As printing evolved, the equipment became increasingly simpler and more portable, with platen presses, phototypesetting, and eventually, desktop publishing.

Fast forward to today, and literally *anyone* can write and publish a book, pop it on Amazon, and achieve worldwide exposure, with potential audiences in the **billions**.

The business benefits of GUIs

Of course, technological development isn't natural selection. Machines and software don't evolve of their own accord. There's always a business case behind such developments, and in the case of the arc toward GUIs, the business case is simple:

The more people that can perform a task, the more efficiently and quickly it can be performed at scale.

Today, web development is an incredibly specialized task requiring years of training, mastery of a broad set of tools and languages, and an ability to keep up with a technological landscape that changes literally daily. And marketing isn't much different. The latter discipline has benefitted from a broader swath of no-code tools, but the knowledge required to be effective in the space is still vast and ever-changing. Just think about how often Google changes how SEO works — often at a fundamental level.

That's why we set out to create this book. To help those looking to experience the benefits of no-code for their own business, on both the web design and development *and* the marketing sides of the process. Later in the book, we'll dig deep to show you how to succeed specifically with Webflow's design and content management tools, but we've also aimed to make this book as broadly useful as possible. So even if you're not ready to dive into Webflow, we think you'll find value here. And if you don't, we'd love to hear how we could do better!

With all that out of the way, let's talk about the benefits of the no-code approach.

CHAPTER III

Why no-code?

Business benefits of ditching code for web development and marketing.

— John Moore Williams, Senior UX content strategist

> *It's easier than ever to build software.*
>
> *Now, in just a few hours, you can create something that used to take specialized skills and countless days.*
>
> *This is all because of the "no-code" tools that let you do so much without writing a single line of code.*
>
> — Hiten Shah,

Now that we've positioned the "no-code" movement in its historical context, it's time to dig into the why of ditching code for website design, development, and marketing.

Easier prototypes & MVPs

Initial prototypes of a digital product often don't need anywhere near the engineering investment as their initial launch counterparts will. Early on, something as simple as a series of well-designed images (like an InVision prototype) might be enough to communicate the core idea to potential investors, early adopters, and future team members.

As the idea evolves, more and more fidelity will be required — but even then, tools like Webflow, Bubble, Glide, and Voiceflow can provide incredibly robust experiences that could be more than enough to pique interest and validate hypotheses. And once you're ready to start exposing your idea to the public,

Webflow, Bubble, and Carrd make for beautiful, highly efficient tools for the landing pages you'll need to communicate your core value proposition and other benefits.

Fewer dependencies

Everyone who's ever managed a project, regardless of discipline, knows that the more people you need work from, the more difficult the logistics of on-time and on-budget delivery get.

The traditional digital delivery process relies on *at least*:

1. Content
2. Design
3. Development

And for more complex deliverables, engineers may be needed as well. Plus, each of the three disciplines above may well involve multiple practitioners, so the above list could well look more like:

1. Content (strategist, copywriter, SEO, information architect)
2. Design (UX designer, UI designer, interaction designer, animator)
3. Development (front-end dev, content dev, etc.)

And we're not even addressing research, which should really precede content. We're also leaving out all the stakeholders and subject matter experts you might need to involve to ensure that content and design are delivering the right messaging, visuals, and overall experience!

No-code tools' value lies in the fact that they can remove the developer and/or engineer from many project workflows. With the right tools in hand, the content strategists and/or writers can model their content in a usable database, not just a diagram or spreadsheet. The designer (or visual developer) can not only mock up and prototype a solution, but actually *build it* in functional HTML, CSS, and JavaScript. Or vice versa!

This isn't to say that projects work better without developer or engineer help. Or that they're in any way better without those professionals. But by freeing them from having to work on marketing assets like new landing pages, blog builds, etc., you give them more time to work on other, more complex projects. Projects where their value is truly realized.

Faster path to launch

By freeing your marketing team from dependencies, you also make the path to launch faster and easier. Instead of pulling a dev from product work, your designer can implement and

hook up any required forms. Instead of pulling an engineer to create and connect a database to your dynamic content pages, your content strategist can handle the modeling and structure. And your paid marketing team can launch new ad campaigns backed by custom landing pages they built themselves. The fact that smaller teams can move faster than larger ones isn't a wonder — it's an intrinsic truth.

To be clear, this isn't to say that you can skip important steps like research, subject matter expert interviews, or stakeholder reviews. These steps may slow down the process, but they're indispensable, and they can often be done in parallel with other work.

Lowered production costs

Developers and engineers are typically — by dint of their extensive training, key role in product development, and relative rarity — more expensive than your average marketing team member. Fair or not, it's a reality. So any time you can remove a developer or engineer's time from the project equation, you're not only saving time, you're saving precious funds.

More autonomy for your marketing (team)

With lowered costs and dependencies comes greater freedom to take risks. To try things that might not ultimately work — or scale. But even failure represents a valuable learning opportunity.

For example, the Webflow team recently began experimenting with various new-user onboarding formats. With a focus on learning over winning, we were able to try solutions that we hypothesized wouldn't work — but even if they didn't, we'd at least *know* that approach X wouldn't work. Because we'd actually given it a shot.

And you know what? Our initial test *did* work. In spite of our hypotheses, which were informed by both quantitative user data and qualitative user interview findings, *both* versions of what we assumed would be failed tests provided statistically significant improvements in active time in the Designer, the key metric we were aiming to move the needle on.

You really do never really know until you try. And now that we've tried, we know.

The obviation of lower-value toolsets and disposable artifacts

Up till the present moment, the design process for digital products and websites has followed a pretty familiar path:

Idea —> design

Someone — a designer, a product manager, a content creator — has an idea. They begin to flesh the idea out in the format that makes the most sense for them, be it a Dropbox Paper doc, a piece of real paper, or a design tool like Sketch. The creator likely iterates there for a while, then reaches a threshold of confidence in their idea: the point where they're ready to share it and get feedback from others.

This is where the most popular design tools begin to lose their relevance (at least, up til late 2018). Because they're largely static: you can share them with others, but they can't really *interact* with them in any meaningful way.

Design —> prototype & collaborate

Of course, the makers of these design tools have realized this hard limitation, and began to build tools like Sketch Prototyping, Adobe XD, and InVision to bring those static mockups to some form of life.

But for almost all of the above tools, the prototype is itself a disposable artifact. People can interact with it and provide feedback, but that's where its usability ends, because it's really just a series of images cunningly linked together to fake a real, interactive experience. They have no usable output beyond the prototype, collaborate, and iterate stage.

Build —> scale

It's here, where the rubber meets the road, that design tools have historically ceased to be of much utility. Developers and engineers pick up where the designers leave off, working largely to translate the static designs they were handed into real, functional code.

This is also where most marketing teams' and makers' start feeling their dependencies. The new site, blog, or landing page leaves their hands entirely and becomes subject to unpredictable shifts in the larger product roadmap, where dev/eng "resources" may suddenly be reassigned, run into tech debt issues, or other bugs. Marketers and makers are left to sit and wait (and move on to the next idea).

The moment for “no-code” tools

This is exactly where “no-code” tools — and in particular, Webflow — really come into their own. With the lowered barrier to entry these tools represent, marketers and makers can ditch the dependencies and get something out into the wild that one precious day, week, or month earlier — and start gathering feedback, iterating, and learning faster.

All of that also means that static “design” tools — in the broadest sense — become less and less useful. Which means fewer disposable artifacts — and lowered software costs.

Last, but most important: a more diverse cast of makers

It’s no secret that Silicon Valley has a bit of a diversity problem. The vast majority of the startups that have gained real traction are therefore designed to meet the needs and overcome the “challenges” of people who are mostly like their builders.

That is: white, male, middle to upper class, tech-oriented, etc.

By diversifying the array of people capable of building and launching digital products, no-code tools have the potential to also diversify the range of products being built and the speed with which they can come to market.

To date, the tech landscape has been dominated by those with computer science degrees. Engineers. People who are trained to approach and overcome problems in a particular way. People who value efficiency, data, analysis.

Those with backgrounds in the humanities tend to think of problems differently. Efficiency is rarely much of a concern. Quality, affect, and experience tend to dominate their thinking.

So you have to wonder: what would a comp lit major build, if they were to build software? How would a philosopher, or environmental lawyer, tackle technological challenges?

For the record, I realize that the above includes *many* generalizations. Our own CEO, Vlad Magdalin, trained as a 3D animator before switching over to engineering. And it's hardly as if there are no non-white makers out there.

But we *need* more. More diversity of background, viewpoint, and opinion. Because the more diverse our cast of makers becomes, the more diverse the solutions available, and the more holistically those solutions will meet contemporary problems.

CHAPTER IV

How to validate startup concepts with no-code

New products are useless if they don't fill a need. With Webflow, you can quickly prototype and test your idea without a massive investment.

— Cameron Roe, Digital Marketer

42% of startups fail because they don't solve a big enough market problem. Take the time to validate your idea and increase your chances of success — the kind we've seen at Dropbox, DoorDash, and Robinhood — with the help of Webflow.

Promoting a new startup can feel a lot like surfing. We all want to ride the wave of the latest trend ...

And to do it without blinking an eye, making everything look effortless.

The problem is, most startups end up underwater.

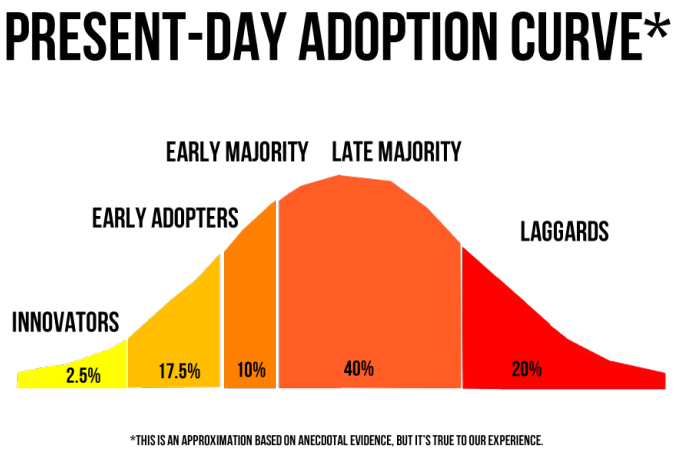
Startup founders get excited about new ideas, but often fail to effectively communicate and validate those ideas with real data.

Why validate?

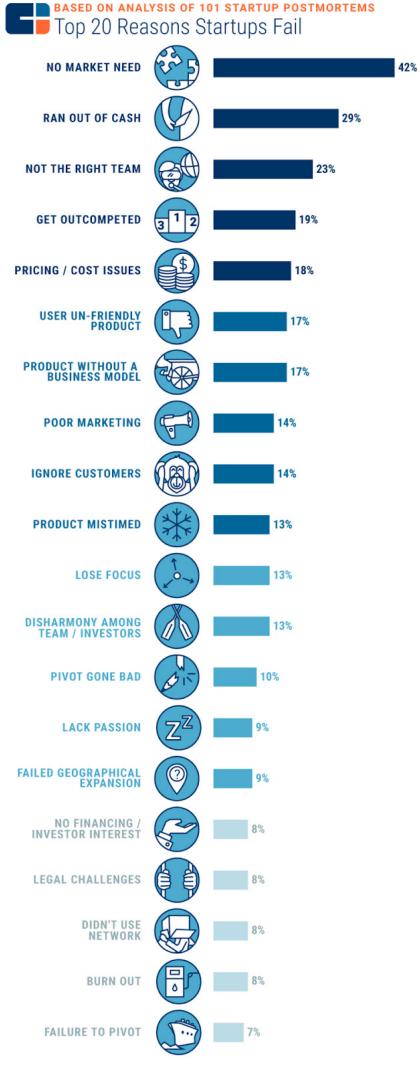
In business, cool ideas are easy — everyone seems to have them. But what really matters is having customers.

Most people aren't willing to buy into a new product until it's been thoroughly tested, proven, and widely used.

If we look at a market adoption curve showing how a new startup enters a market, we see that 40% of the market won't use a product until the early majority is using it. Social proof is a big reason among many.



CB Insights did a report on the Top 20 Reasons Startups Fail and the number one reason at 42% is no market need.



There will always be a lot of initial assumptions in the early stages of a company — but it's important to continually question and validate those assumptions with real data.

For example, you might ask questions like:

- Who is the ideal audience?
- Where is that audience?
- What is the value proposition that appeals to them?
- What motivates people to engage with our brand?
- What drives brand value and repeat purchases?

Once you understand what questions need to be answered, you can map those questions to a customer journey and focus on validating the most important question:

How can I create a customer?

When you can answer this, you'll have validated your startup.

Validate like Dropbox, DoorDash, and Robinhood

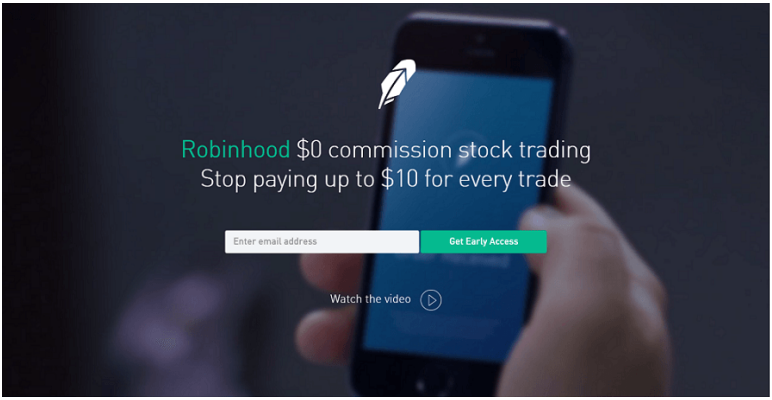
All the top startups in Silicon Valley had to validate their assumptions. They all had to answer key questions about their audience, value proposition, and converting leads into paying customers.

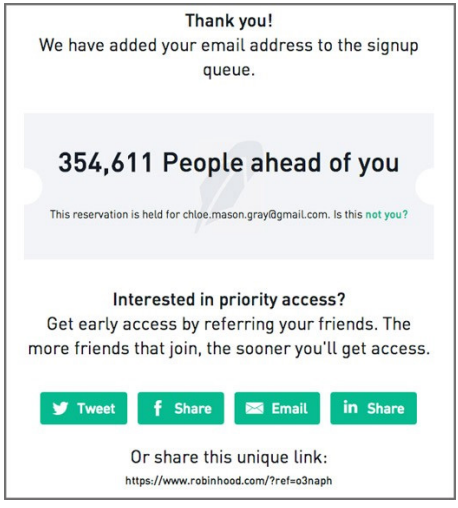
Each of these successful companies started with a simple landing page, ran experiments, and gained valuable feedback about their idea. These are critical steps for a founder to tackle.

At Dropbox, they created a simple landing page with a demo video and a sign-up form to collect emails. They wanted to test if potential customers would be interested enough to share their email and learn more — that is, find out if there was "market demand."

Robinhood set up a landing page to highlight their key differentiator — $0 commission stock trading. They included a viral share option and rapidly grew their email list before they even had a product.

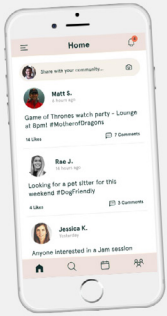
This helped them build a massive email list and create a sense of community that became an audience they could test with.





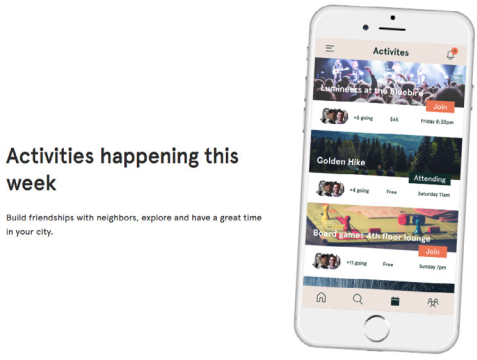
I helped a client validate their startup idea by setting up a landing page that included a sign-up form and outlined their value proposition and key benefits. We added screenshots of the app's features to gauge interest.

The Social Layer for your Apartment Building



Connect with your neighbors

You now have an easy way to start conversations and make plans with your neighbors.



While every startup will ask slightly different questions, they all take a similar approach to defining their assumptions, creating new experiments, and testing those experiments to validate or invalidate the outcome.

Define the audience

Before you can solve any problem, you need to understand it. This almost always starts with understanding your customer/market/audience — a group of people who share the problem or set of problems you're trying to solve.

To define and understand your audience, it's important to consider them from a few different angles.

Demographics

What are the characteristics of your audience? This should include things like:

- Gender
- Age range
- City
- Job title
- Marital status
- Level of income
- Level of education

For a business-to-business (B2B) market, you might decide to target lawyers in New York City, ages 30–45, who earn an annual income of 130 thousand or more.

For business-to-consumer (B2C), you might target single people in Boston, Philadelphia, and New Jersey between the ages of 21–35 who are studying for a bachelor's degree.

Interests

Another way you might want to define your audience is based on their interests. This is a common and highly effective

targeting method on Facebook — it shows what brands your audience already engages with.

For our B2B example, you could specify lawyers interested in specific software companies, like Apple and Google. They're likely early adopters of technology vs legacy software users.

For the B2C example, you could reach computer science and business students by targeting people interested in specific schools, like MIT and Harvard. Or you could target competitors to reach the same type of product affinity.

Website analytics

A final way to define your audience — which will become important when you advertise with remarketing campaigns — will be your website analytics.

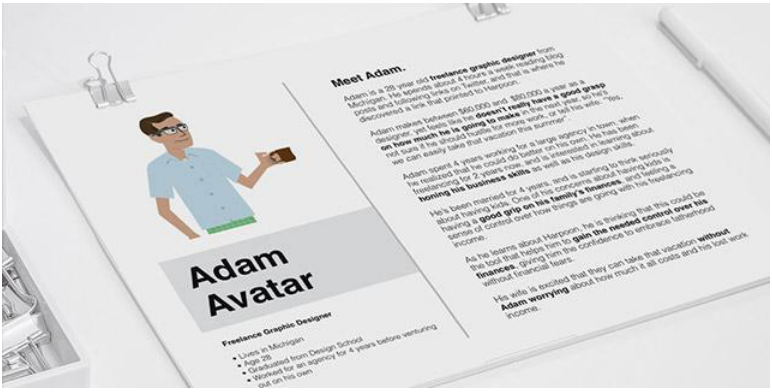
This includes tracking metrics like:

- Acquisition channels (where they came from)
- Devices used
- Number of pages visited
- Session time
- Bounce rate
- Completed conversions

This data can be used to define and remarket to audiences who've interacted with your website in specific ways — pushing out additional content that creates urgency and closes a sale.

Creating a customer persona

Once you have enough data on your target audience, you'll want to create a customer persona to outline the profile of your ideal client in a single document, sometimes know as a "source of truth." This will be your initial set of assumptions you'll use to test and find product/market fit.



Model experiments in Webflow

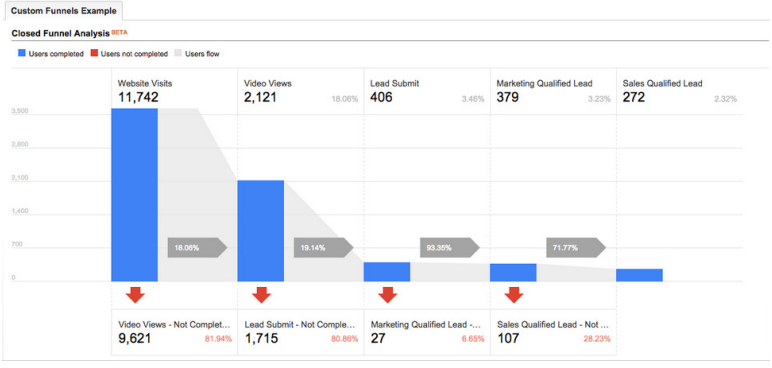
Creating a new customer is hard work. It requires a lot of experimentation and testing to get right. Creating a customer is

really about cultivating a relationship — a series of engagements strengthened from multiple points. Each point builds the customer relationship, develops trust, and helps them make decisions about your product.

Almost everything you'll do with your content marketing strategy is meant to strengthen the customer relationship and drive users closer to completing a goal. In marketing, this is known as a campaign.

Sometimes the best campaigns are experiments that turn out to be successful. Experimenting with your campaigns can help you stay creative and focus on what serves your audience.

Webflow makes it easy to model new campaign experiments for each stage of the customer journey. If you were to view a funnel report in Google Analytics, you might see something like this:



To create the right type of content across each stage of your funnel, or "customer journey," you'll want to consider the stages a buyer goes through.

Here's a quick breakdown of the stages of the funnel, and the types of content typically associated with each:

Awareness

The awareness phase is when a user discovers and engages with your brand for the first time.

In this stage, you'll want to drive site visits through blog posts, ebooks, guides, or checklists.

Interest

Once the user has interacted with your brand in the awareness phase *and* decided they want to know more, they move into the interest phase and become a lead.

During this stage, you'll want to collect leads with webinars, case studies, and email courses.

Conversion

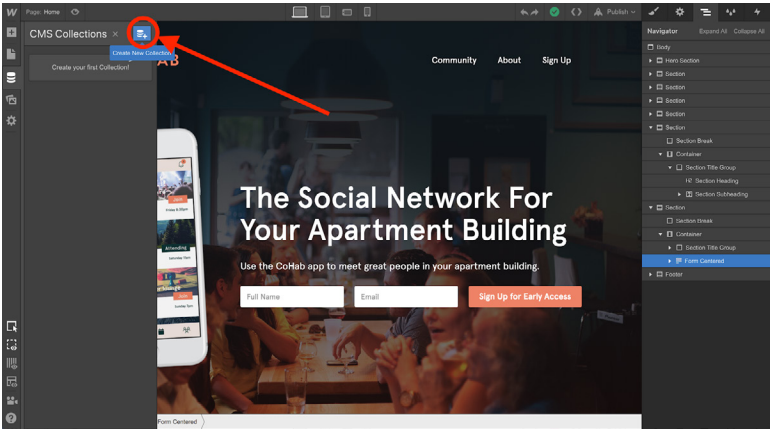
Finally, once a user has been engaging with your brand consistently and trusts your offer, they'll reach the conversion stage and make a purchase.

In this final stage, you'll want to drive purchasing behavior through consultations, coupons, and free trials.

Modeling campaign content types in Webflow

Once you understand what type of content will move users through a customer journey, you can begin to model those content types in Webflow.

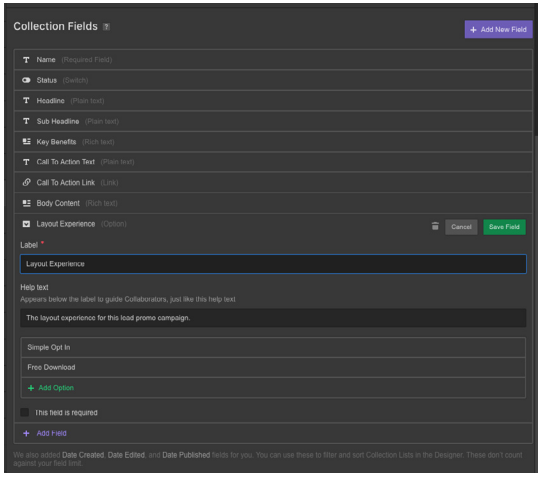
For example, let's say you wanted to model a campaign Collection to create new landing pages for those campaigns. You would create a new Collection from the CMS Collections tab and name it "Campaigns."



You would then include the common data attributes of your campaigns which might include:

- Headline
- Subheading
- Key benefits
- Call to action
- Call to action link
- Body content

You could also use Webflow's option field type to create a set of options for a layout experience (more on this below). This would allow you to adjust your design using conditional visibility and create multiple experiences to test with.

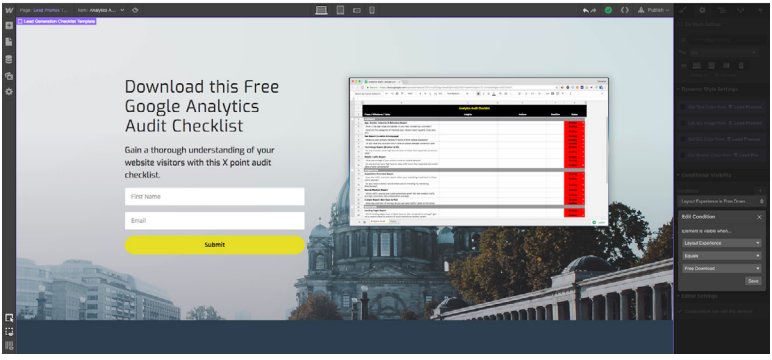


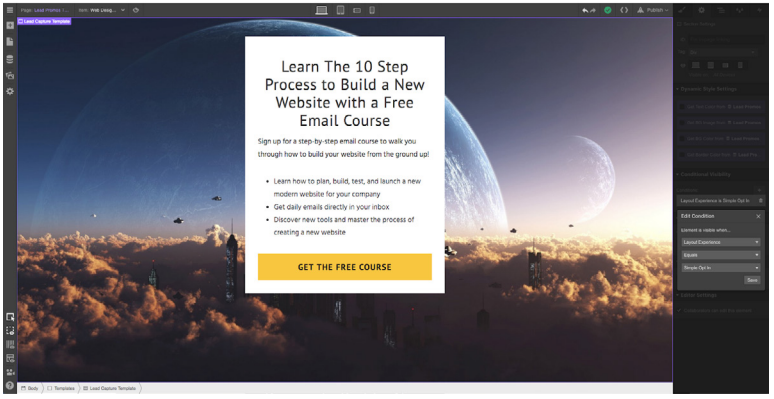
Design campaign experiments

When you design campaign experiments, you'll want to design multiple layout experiences to test assumptions for each campaign. Layout experiences should be based on the message you're promoting and the stage of the customer journey you're targeting.

You can think of your campaign templates as the container for your data. The data is the content and the layout is the container that provides the experience for the user.

For example, let's say you created 2 new campaigns — a checklist and an email course — to drive awareness and leads.





By selecting the layout experience option you created, you'll be able to render your data in an entirely new way. This reduces the need for additional Collections and helps you speed up experimentation.

Optimize and validate experiments

To create the best customer experience online, you'll need to review analytics and iterate based on what you learn.

For example, when reviewing your Google Analytics reports, you should ask critical questions about your data:

The audience report

- What's the age of the audience?
- What's the gender of the audience?
- What country and city are they coming from?

The channels report

- Where did the traffic come from?
- What was the bounce rate?
- What was the conversion rate?
- How many conversions were generated?

The funnel report

- How far did they get through the funnel?
- Where did they drop off?

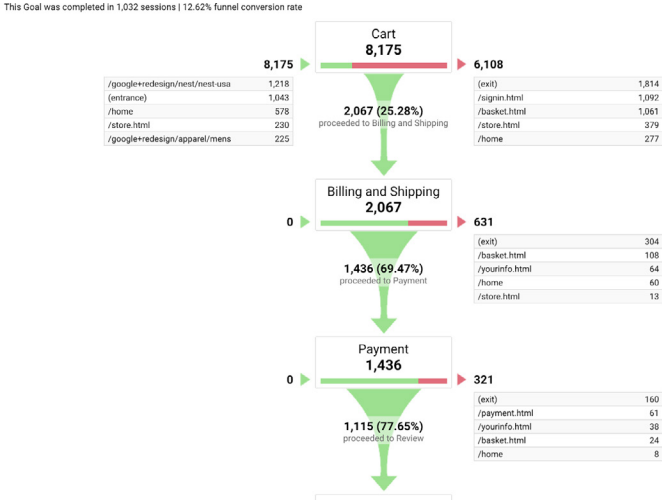
These reports will give you key insights about your audience and help improve your campaigns — Google Optimize is a great tool for this.

You can set up A/B tests to make changes like:

- Updating the copy
- Adding new media
- Changing the targeting
- Updating landing pages with high bounce rate

Over time, you should begin to see distinct audience segments respond to specific campaigns and perform better than others. And you can have confidence that your assumptions

are indeed proven (or subverted!): a specific audience, with specific campaigns, and a specific experience will produce a desired outcome — usually sales.



The tl;dr (too long; didn't read)

Creating real value in a market is full of changing dynamics: the existing supply (competition), trends, user needs, trust, and, of course, the market adoption curve.

If you truly want to build a startup, you absolutely must validate your assumptions. Webflow makes this easy to do with the right Collections, the right content, and an engaging experience.

When you focus on your (potential) users, gain their trust, and sell them your product, you've validated your idea.

> *A good way to know whether or not you have reached product/market fit is if at least 40% of your users would be very disappointed if they could no longer use your product. Make things people want.*
>
> — Sean Ellis

CHAPTER V

Enhancing customer experience with no-code

The great thing about no-code tools today is that they empower you to focus on providing a more personalized customer experience.

— Jeff Cardello, Content Creator

No-code means putting your budget on building memorable customer experiences, rather than focusing heavily on infrastructure.

Bryant Chou, CTO and cofounder of Webflow, spoke at No-code Conf 2019 about where things are headed and how this liberation from managing the backend will change how brands connect with their audience.

In 2009, Chou was a junior engineer working at Intuit. His job effectively came to a standstill as the company went into a state of reorganizational limbo. During this idle time, he set out to create an app based on cutting-edge technology for the time — Android 1.5.

Inspired by his childhood love of chatting with his friend on walkie-talkies, he decided to create an app based on this immediate two-way communication. After diving in and learning the ins and outs of building an app on Android 1.5, Chou finished it and aptly titled it Hoot. With a simple interface of one red button, you could record and zap out a message to anyone on your contact list.

Chou put out this app for free and gained quite a bit of traction, cracking the top ten of trending free apps in the Google Play Store — reaching 10,000 downloads in a month. Naturally, he was excited about creating a viral hit app — that is, until

he received a bill from Amazon Web Services. His feelings of elation over the app taking off were crushed when he learned he owed $300 for the infrastructure that made it possible.

> "I *could either spend a lot of time on this app making it better because it was so raw and had a ton of bugs, or I had to scale this server to make it more performant. And essentially I had a tough call to actually shut it down.*"
>
> — Bryant Chou

He made the hard decision and pulled the plug on Hoot.

Infrastructure used to be complicated

Just 10 years ago, you had to know how to scale databases if you wanted to run a popular website or app. And if you didn't, you had to hire someone who did. Keeping things updated and running smoothly required continuous effort, taking up a lot of time.

Today, you don't have to concern yourself with what happens in the backend. Webflow takes care of security, maintenance, and scalability to keep things running so you can focus on what's important — the customer experience.

Webflow means protection

There are so many bad actors out there ruining the internet. Webflow keeps you protected from a wide array of threats that WordPress is especially susceptible to, including:

- Brute force attacks
- SQL injection
- Malware
- Cross-site scripting
- DDoS attacks
- Vulnerabilities from outdated versions of WordPress or PHP versions

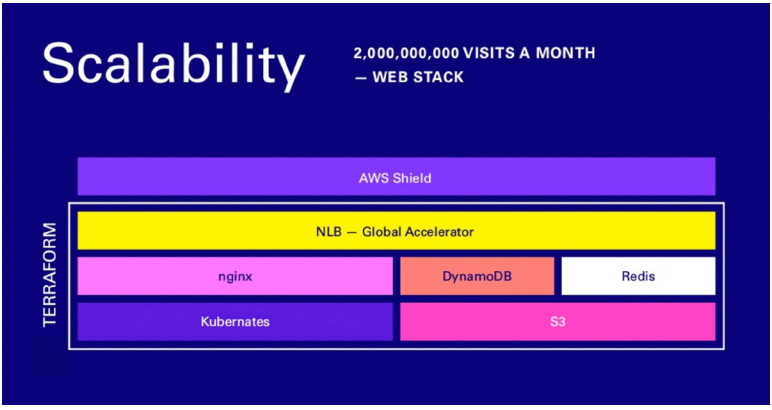
Webflow also does regular penetration testing, which is a controlled way to execute a fake cyberattack to see how security measures are holding up. Being liberated from dealing with these things on the backend gives you more time to put effort into other areas.

> *"Makers can build the best possible products and work on things that actually matter like marketing, growth, leveraging the data that you're collecting, as well as automation."*
>
> — Bryant Chou

Bryant Chou on stage at No-Code Conf 2019

Webflow makes scalability automatic

Webflow's network of websites receive over 2 billion views a month. Let that sink in. That's a lot of page views. Some web hosting providers would get bogged down with this colossal amount of traffic — or worse, go offline. But because Webflow's hosting speeds are like a bullet train, loading times are extremely fast and they can easily scale up to accommodate surges of traffic.



Webflow uses modern technologies for absolute scalability, including:

- Amazon Web Services and their DDoS protection layer shield
- NLB Global Accelerator for load balancing
- Kubernetes to manage the server side
- Amazon S3 for data storage
- The database service DynamoDB
- A specialized version of NGINX that powers 10% of the internet

> *"It doesn't make sense running your own marketing site, company blog, or ecommerce store anymore when the intricacies are taken care of by Webflow."*
>
> — Bryant Chou

How does Webflow do it? Webflow spends $125,000 a month managing, maintaining, and updating its infrastructure, and has a team of engineers and experts staying on top of things. And it works. With 99.999% uptime and an average page load time of 400 ms, it's stable and quick.

The no-code movement lifts the weight of having to deal with software, servers, and other technical aspects of running a website. It lets you focus on putting effort into marketing and other avenues of growing your brand.

No-code frees you to market better

> *"As a late millennial myself, we're not necessarily a generation who responds to sales well."*
>
> — Bryant Chou

The marketing landscape has changed, and to reach this younger generation brands need to build more than a digital platform of bland self-promotion. They need to put something on the web that has depth. To connect with this younger demographic, and to provide a better experience for everyone,

the focus needs to be on the customer. When you don't have to worry about code or what's running the backend, you can give customers an optimal customer experience.

Customer experience supercharges traditional marketing and brings momentum with greater sales and growth of repeatable product lead growth. Products, along with data and automation, are the fuel that drives stellar customer experience.

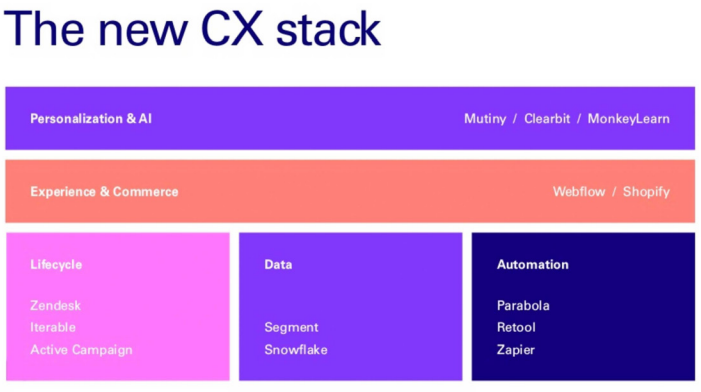
How customer experience will transform the future

> *"A renewed focus on customer experience that involves growth, product, and marketing can really empower modern-day sales and marketing motions so that we can actually build companies through product lead growth."*
>
> — Bryant Chou

Customer experience marketers' expertise encompasses product, content, and growth marketing. They know their audience and work with marketing teams to develop a content strategy and deliver content that their audience will find valuable. They collaborate with visual developers — another important role in customer experience — to craft this content.

The customer experience stack

The customer experience stack makes it possible to offer an individualized and consistent experience for each person navigating through a brand's digital space. Data shapes what they see, and how they interact. Each layer of the customer experience stack comes together in fostering more personal connections with a brand's audience.



Personalization and AI

This evolution of marketing requires a new approach to the customer experience stack. Personalization and AI occupy the top tier of this reimagined customer experience stack. User attributes and data are gathered and integrated into

different applications, like Clearbit or MonkeyLearn, so each person using a website has a personalized experience.

Experience and commerce

Experience and commerce occupies the next level down the customer experience stack. This is where a brand gets the chance to make a good impression and where an audience can interact and buy things from them.

Life cycle

The type of messaging, when it's delivered, and the channels that deliver it matter in making sure that customers get the right information when they need it. Tools like Zendesk for customer service, Iterable for engaging across different platforms, and ActiveCampaign for marketing automation can help facilitate this communication.

Data

This level of the customer experience stack involves data — both where it's kept and how it can be accessed. Tools like Segment, which catches data and transports it to other software, or Snowflake, which compresses down and structures data, are helpful in managing data.

Automation

This level of the stack mechanizes all of the personalized data and transmits it to tools like Parabola, Retool, and Zapier, where data can be analyzed and utilized.

Customer experience is multifaceted

> *"That's what, in my opinion, is the exciting thing about customer experience in the future: where you're really thinking about customer experience from a holistic standpoint and using the best-in-class tools to provide it."*
>
> — Bryant Chou

A website is the focal point in a brand's circle of marketing. The information and data a customer shares gets shuttled off to connected platforms like Zapier, HubSpot, and Salesforce, further enhancing the customer experience and helping a brand better serve their customers.

No-code ushers in a new decade of customer experience

No-code isn't meant to sleight those who have the expertise and passion for all that is data, servers, and infrastructure. Webflow depends on these noble tech geeks to keep things running for the thousands of websites that it hosts.

Why worry about these things when you don't have to? Keep things simple so you can jump on the swelling wave that is customer experience and build a better brand.

Let us know in the comments below how you're using no-code tools to create better customer experiences!

CHAPTER VI

A guide to bringing no-code to your company

Learn how Getaround's marketing team migrated their website to Webflow, improving their web performance and in-house designer team agility.

— Camille Esposito, Brand Designer

In 2019, I co-led the project to migrate Getaround's marketing website from traditional development into Webflow. The migration was ultimately a big success, unlocking not only better web metrics but also unprecedented agility for our team.

Getaround's marketing site before Webflow: lean resources, too many priorities

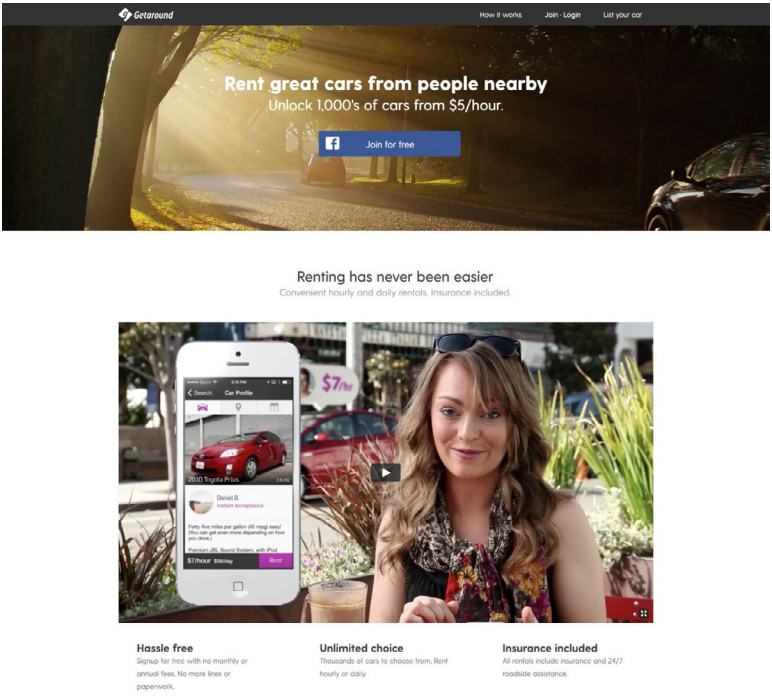
Before migrating to Webflow, our marketing pages were coded the traditional way and owned by engineering. So any time we wanted to make a change to our homepage, we were up against competing product priorities and vying for precious engineering resources.

As Getaround rapidly grew, so did our web goals, which included:

- Increasing organic search traffic
- Converting more users than ever before
- Expanding our evolving brand into our online presence

At the same time, rapid growth meant that our product *also* needed to scale and evolve.

Making our product awesome was, logically, the top priority for our engineering team. As a result, marketing and design updates to our website fell to the wayside. So, in 2018, we found ourselves with a website that looked and behaved ... a lot like a website from 2011.



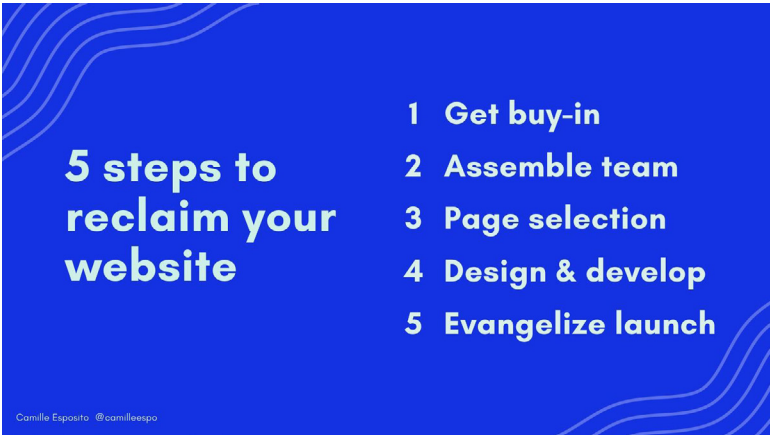
getaround.com, 2011 edition

We migrated to Webflow to change that. Let's dive into how your team can do the same.

5 steps to reclaiming your website

Reclaiming your marketing website with Webflow takes strategy, and we found a lot of success with these 5 steps:

1. Get buy-in
2. Assemble your team
3. Select pages
4. Design and develop
5. Evangelize launch



1. Get buy-in from key stakeholders by strategically "wading in"

If your team or company has never heard of Webflow — or doubts its robustness or efficacy — invest some time in education and demos. Take advantage of Webflow's visual nature to *show* what it can do, rather than relying on discussion alone.

Consider screen recording yourself building a small page in Webflow and using that recording to show just how fast and easy it can be. Use that Webflow page to wow stakeholders with live demos of whichever features your team will most benefit from, whether that's quick copy and image edits using the Editor, or building with the CMS.

In addition to your demo project, share case studies and anecdotes from other companies who made the switch to Webflow — bonus points if you find a testimonial from a team that's similar in size and goals to yours. Stakeholders are more likely to buy in on a tool that's tried and tested, so do your own trial in addition to getting validation from other teams' tests.

Some Webflow success stories for fodder:

- HelloSign case study
- Zestful case study
- Kisi's SEO case study

Hopefully that gets you the green light to move into a trial run of the tool using lower-risk web pages. Think about which pages are (relatively) low stakes for your company: avoid your homepage or any primary driver of conversions or other key metrics.

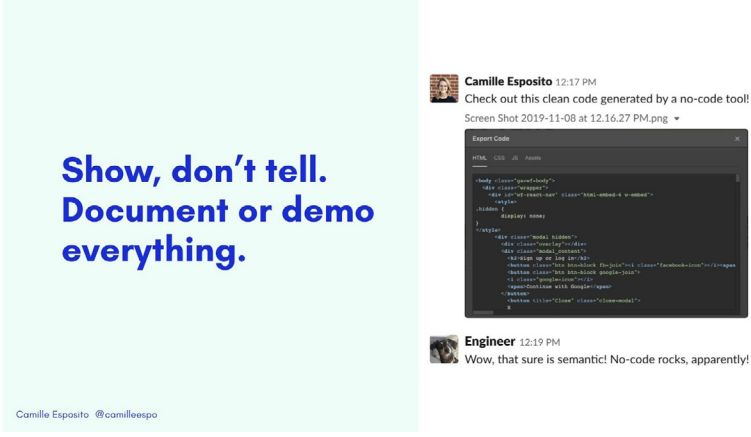
At Getaround, we started using Webflow for smaller landing pages, like promotions and local partnerships.

This gradual wade-in serves as a proof of concept for Webflow, exposing key stakeholders to its capabilities (and its limitations) without the pressure of massively impacting crucial metrics like sign-ins or conversions.

During your trial run, demo relevant features to the appropriate stakeholders to get them excited about Webflow's capabilities.

- Show engineers how clean and semantic Webflow's exported code is
- Dive deep into Collections with content creators and managers to show how they can build, manage, and expand their own custom content structures

- Show copywriters how easily they can edit content on a page via the Editor
- Have designers play around with Interactions or build page elements quickly and seamlessly in the Designer



Use the wade-in to prove your claims so that your team can move on to migrating the beefy stuff into Webflow. At Getaround, our claims included the following:

- Quicker production time
- Smaller required team
- Equivalent or improved business performance metrics like conversion rate
- Improved search performance

The Getaround wade-in was a success. Our small team of 2 non-coders built and launched fully functional web pages within a few days. We didn't hurt conversions, and after the initial launch, we launched more iterations in one week than we had in *years*. We were also able to iterate on SEO, resulting in improved organic search traffic.

By now stakeholders trusted the benefits of Webflow for both our team and our web metrics, clearing the way for us to continue the migration.

2. Assemble your team

Once you've gotten buy-in on the wonders of Webflow, it's time to assemble your web team.

One of the most powerful lessons I learned from Getaround's migration to Webflow is the importance of educating teammates that "no-code" does *not* mean "no web squad."

Without investing in a proper web team, your team will not fully reap the benefits of no-code.

My coworker Meg and I handled almost all aspects of our migration to Webflow. Between the two of us, we:

- Designed *and* developed pages
- Built guidelines and systems
- Managed developer contracts
- Coordinated with engineering on cross-functional needs
- Came up with our migration strategy
- Launched pages
- Wrote copy
- And more!

But just because something *can* be done doesn't mean it *should be*.

Our small team combined with our ambitious timeline meant that we sometimes built things that were fast and dirty: fine for the sake of meeting our launch deadline, but ultimately leading to technical debt and more work for our future selves.

To avoid that, put together a team much like you would for a website built the traditional way.

The question of who should comprise this squad is an interesting one, as no-code is only just gaining major traction in the tech world and roles continue to form and evolve, but here are the roles I would have liked to have on our migration project team:

- ***Product manager*** *to own strategy and cross-functional work*
  Huge bonus if this PM can bridge the gap between Webflow and traditional dev, especially if your migration involves shared functionality with product, like a navigation bar that keeps a user logged in across multiple environments.

- ***Project manager*** *to keep all the moving pieces on-track and see that projects get done*
  Your product manager could potentially serve this role as well.

- ***Visual developers*** *who are Webflow experts and know development best practices*
  This is crucial for the scalability and maintainability of your Webflow site. Almost anybody can cobble together a site in Webflow, but it takes development expertise and experience to build it in such a way that it will scale and evolve cleanly, without accruing technical debt.

- ***Web designers*** *with Webflow familiarity to create mocks and flows*
  Designers should at least familiarize themselves with Webflow so that their designs both take advantage of Webflow's awesome capabilities (like Interactions) and work within its limitations (which are, in large part, the limits of the web)

- ***Copywriters and SEO experts*** *to create on-brand, performant content*
  Be sure to work closely with both writers and SEOs to ensure the page structure works well for the content and for optimization!

Be transparent about every task that needs to happen in your migration, from content mapping to page selection to product collaboration, to prove the need for this squad. And remind your team (and yourself!) that you're still setting up a robust website. Even though you're doing it through a no-code interface, you're still essentially coding and building complex functionality.

3. Select pages to migrate to no-code

Now that you've gotten buy-in on Webflow and assembled your dream team, figure out which pages or chunks of your website make sense to move into no-code.

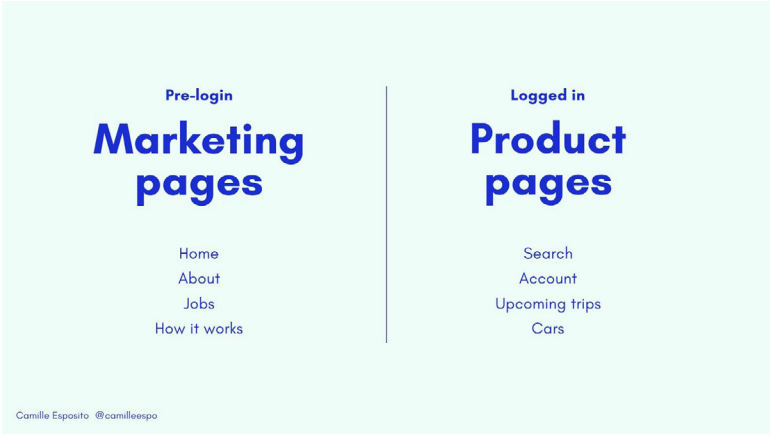
This will be different for every team and company. Keep in mind:

- **Dependencies**
  Is your team responsible for increasing conversions on a landing page, but your engineering team has to execute that work? That landing page would be a great candidate for migration.

- **Any product functionality that would require extensive custom code in a no-code tool**
  A product page that has complex logic or API dependencies may not be a good candidate for migration. Extensive custom code defeats the purpose of a tool like Webflow, and greatly reduces your ability to enjoy its benefits.

- **Scalability and future page goals**
  Any pages that would benefit from Collection lists and the CMS, or could grow to become a collection of pages, should definitely be migrated. Top candidates include: landing pages, case studies, blogs, production education — really any content type where writing is bound to happen faster than design can support.

For Getaround, the breakdown of which pages to migrate was clear. Before a user logs in, they might explore marketing pages like home, about, or how it works. After logging in, they're in the product — searching for cars, booking cars, etc.



Since marketing is responsible for the performance and goals of pre-login pages, we migrated those to Webflow. This also meant that engineering could fully focus on our product, without frequent interruptions from marketers and designers to replace hero images or update H1s.

4. Design and develop with your no-code medium in mind

Every medium has built-in capabilities and constraints, and you need to keep both in mind as you dive into your deliverables. The following were key considerations for my team.

Consult your visual developers early and often

Check in with your visual developer frequently to make sure your ideas can be built scalably and with Webflow's best practices in mind. Most everything is possible, but you may need to make some design modifications to avoid relying on extensive custom code or hacky solutions that won't scale well.

Become a no-code expert

Take advantage of no-code resources like Webflow University, Makerpad, and Design+Code.

If you're a learn-by-doing person like I am, building sites with the help of these resources is a great way to immerse yourself in the world of no-code and familiarize yourself with all of its possibilities and best practices.

Maximize your use of the CMS

Webflow's CMS is one of its most powerful features for a fast-paced marketing team. Again, work closely with your visual developer to identify pages and sections that can be built with Collections. This will allow your team to launch dozens of pages with only slightly more effort than one page.

Use visual development guidelines

You're still building a professional-grade website here, so you need clear and robust development guidelines for things like:

1. Naming conventions
2. Global styles
3. Usage of components

Be cautious with custom code

Relying extensively on custom code defeats the purpose of using Webflow: you'll see big chunks of nothing in the Designer UI rather than seeing exactly what you'll get on publish. Maintaining various snippets of custom code also quickly becomes unwieldy.

Do use custom code for:

- Minor CSS styling that can't be done through the Designer, like changing the arrows on a slider
- Simple JavaScript functionality, like adding logic behind a sign-in button

Avoid using custom code to:

- Build entire elements by writing custom HTML and CSS. It may sound glaringly obvious, but it's an easy trap to fall into, especially when working with engineers who may not fully understand Webflow best practices.

5. Launch and evangelize your no-code site

After launching your no-code site, share the project, its successes, and everything you've unlocked by moving to no-code.

Keep stakeholders bought in and excited with a steady stream of education and demos:

- Send articles to your engineers about Webflow's backend or new API integrations to prove how it's keeping up with changing technologies
- Demo how quickly a designer can update your homepage with a new layout and new assets
- Show how easy it is to generate dozens or even hundreds of pages from a single Collection page

Pro tip: people love a good Webflow GIF.

Most importantly, document and share the goals you achieved because of your no-code migration. These might include:

- Number of page launches per week
- Time to create a new page, from kickoff to launch
- Performance metrics like SEO, conversions, and site traffic

Getaround's marketing site post-Webflow: improved web performance and team agility

Here's what Getaround has achieved in the 3 months following our migration to Webflow:

- Weekly new-page launches
- Keep in mind that we didn't really launch a new page from 2011–2019!
- Total site sessions increased by 25%
- Organic search traffic increased by 15%

Most significantly, we are now much more nimble and able to keep up with the speed of our business. A few months ago, this was pressure tested when a major (and time-sensitive) business initiative needed to launch about a week after the

project was kicked off. We easily designed and built two landing pages to accommodate this in less than 5 days!

Migrating Getaround's marketing pages into Webflow was hugely impactful for our business, upleveling not only our web performance metrics and the quality of our web content, but also the velocity and nimbleness of our team.

Rebuilding an existing legacy website in a no-code tool is no small task, but I hope these steps help bring it into reach for your team: the results clearly pay off!

CHAPTER VII

Planning for the switch from no-code to code

— David Head, co-founder at Sixty

My name is David Head. I've been building no-code products and features since 2014. I led my team to get funding from Y Combinator for the summer 2017 batch with a product that went through two no-code iterations before transitioning to a Ruby on Rails stack.

I'll tell the story of how we built our two MVPs and navigated nine months of 50% per month growth until we finally rebuilt in Rails. Throughout this narrative, I'll describe what I call "no-code limitations" and "stack transition steps," which I'll detail below. My goal here is for you to learn how to be more successful building no-code products and the technology companies that emerge on top of them.

Principles

A core principle I've found in every no-code project is *the more traction you get, the more beneficial it becomes to move to code*. You'll notice three things start happening, which we'll call *no-code limitations*:

No-code limitations

1. Feature limitations: you're limited in how much you can customize features to your specific use case

2. Community limitations: you'll want to hire someone to maintain the product/feature and build more code
3. Infrastructure limitations: there is a lower limit to how many people can be using the product at once

The more traction you have, the riskier your situation will be. You have to plan a smooth and swift transition from your current stack to your next stack. Your next stack may be another no-code stack, but eventually if you continue to be successful, it will usually turn into a code stack like Ruby on Rails. We'll call these *stack transition steps*.

Stack transition steps

1. Predict breakdown: always be thinking of when and where your current stack will break down
2. Build new stack: start building new stack to transition to when your current stack breaks down
3. Transition to new stack: migrate all of your data and users to the new stack

Finally, unless your coding skills are that of a senior developer, plan to hire someone to lead development of the codebase when it gets to that stage. Someone who can handle all technical decisions, rather than a junior developer working under you.

My backstory

In 2015, I owned a web design agency that built a lot of Squarespace websites. We were partnered with Jake Jorgovan, who created a popular course on how to build a Squarespace website. Jake's course generated dozens of people reaching out to ask things like, "Will you help me polish my website's design for launch?" Jake would refer these leads to us.

Most web designers didn't want these small-scope leads since all the back-and-forth made them expensive. But we discovered that working with the client via video conference allowed us to bill 50% more than our standard rate (so, $150/hr) and launch their site in about 2 hours. Which made it worth it for most professional designers.

Thinking about all of the other web tools marketed as do-it-yourself, we figured there was a larger market for on-demand help like this far beyond web design. So I reached out to a colleague, who also owned a web agency, to be my cofounder on this side project.

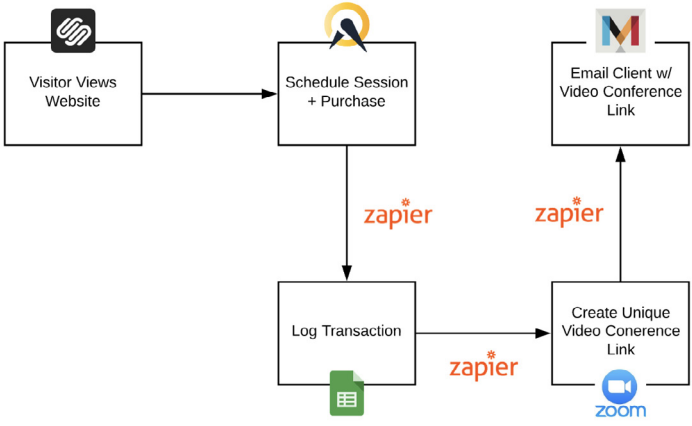
Squarespace stack: proving the concept

My new cofounder handled the sales and recruited web designers to offer their services. I focused on hacking together an infrastructure to scale the service. My goal was to bring

on another team member to lead the tech once we proved the concept and went full time.

Since we had an abundance of leads, our first bottleneck was having an infrastructure to facilitate web designers doing on-demand sessions via video conference. I solved this with the following stack:

- Acuity Scheduling embed
- Zapier to send the transaction data from Acuity to a Google Sheet
- Zoom API plan to set up the video conference
- Mandrill to send transactional email and video link
- Squarespace website



Once we had this stack built, and tested out with a few clients, my cofounder started recruiting designers to join.

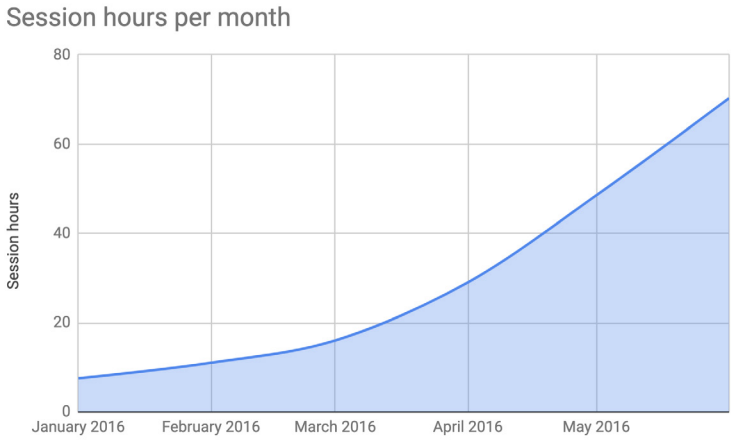
When building the platform and predicting when it would break down (step 1 of *stack transition steps*), I knew Squarespace and Acuity were the first bottlenecks.

1. Acuity had a team account but no API plan like Zoom. All of the designers would login and sync their calendar like they were a team member. This was a dealbreaker for scaling.
2. We had to make each web designer a unique page in Squarespace because there was no CMS. So updating the template of a designer's profile meant updating each page individually. This also wouldn't scale far either.

Other breakdowns included:

- We were unable to add basic web app features one would expect, like user login
- Client users had to enter their credit cards every time they scheduled a session. There would be no record of past transactions they could see in a UI, which was also an opportunity to prompt follow-up sessions
- Designers had to put in requests changes for profiles and we had to change it manually

We scaled on this stack for 5 months at 50% growth per month. Our primary metric was hours of sessions per month.



Going back to the *no-code limitations* concept, our primary issue was #1, *feature limitations*. At the *stack transition steps*, we predicted things would get problematic around May and we'd need to change stacks.

Bubble stack: building a web app

To keep increasing revenue, we needed to make it easier for clients to buy, and increase the number of designers on the platform so there was more time availability. Our vision was to eventually make it possible for clients to hire a designer within 60 seconds.

Our next step was to rebuild most of our stack. We thought we'd need to hire a developer, but there was a new "codeless web app builder" platform called Bubble that seemed promising. It seemed to be able to do everything a traditional web app could, but way faster. We decided to give it a try. Next steps:

1. Replace Squarespace with Bubble so we could build a client dashboard and manage dozens of designer profiles
2. Seed Bubble's database with our Google Sheets data
3. Rewire Zapier to still post data into Google Sheets to use for analytics

Since we were growing revenue at this point, it made sense for one of the junior front end developers at my web agency to learn Bubble so I could focus on other growth tasks. It took him one month to learn Bubble, rebuild our platform, and take us live.

We now had a signup flow, a client dashboard, and had integrated Stripe Connect to store client credit cards. Next, we needed a feature allowing each designer to sync their calendars via OAuth. We assumed that since we were good with APIs, we could make anything happen in Bubble that could happen in a codebase. Of the few tools that seemed promising for this, we planned to use Timekit since it seemed the most polished.

This ended up being a massive learning opportunity for us. When it was time to implement Timekit, we realized there were "callbacks" that had to be programmed for the OAuth to work — they were impossible to hack into Bubble. We now had a new *feature limitation* on a critical feature. We decided to scale around it as long as possible.

Next up was enhancing the client dashboard so they could view past sessions — an extremely basic functionality to program into a web app. When we programmed it in Bubble, there was about a 2 second delay until the historical session data would show up on the screen. While it technically worked, the UX felt janky. We spent a dozen or so hours working with the best Bubble freelancers to optimize this workflow, but there wasn't much of a difference. This was an *infrastructure limitation* that turned out to be a UX issue.

While we worked through all of these issues, each new one became increasingly more challenging. While our developer picked up Bubble and rebuilt the platform blazingly fast, there hit a point where all of us were overwhelmed and looking for a more senior developer to take the lead.

Note: You'll live or die by the fundamentals

Being able to stick apps together to do things is easy. Knowing the fundamentals of web app architecture is where you'll get killed. This is because you're creating technical debt. Not just in

the sense that you're not in a codebase, but because you will do things like improperly model your database.

This is what happened to us. It was easy to spin up the database and associations in Bubble, but because it wasn't done in a fundamentally sound way, the database queries took longer and we would have to change the schema and do database migrations when adding new features.

This is the same with Webflow and Squarespace users who were reaching out to us for help: they could put something together that technically was a live website, but they weren't proud enough of it to make it public. Their problem was they didn't know web design fundamentals.

We searched for a senior Bubble developer. There were only about 10 to choose from on the forums (this was pre Bubble-developer directory). While working with them, we learned that our junior developer who just learned Bubble two months previous was at roughly the same skill level.

Everyone had a similar background to us: they wanted to quickly hack something together and didn't know how to code. No one had experience building scalable web apps and definitely not a Computer Science degree.

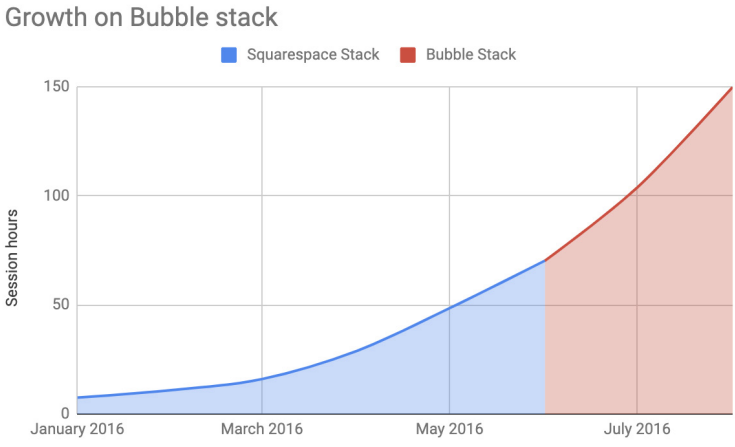
The average rate for a Bubble developer was about $150 an hour, too. Compare that to the average Rails developer, who is

$80–100/hr, has a few years of app development experience, and often a CS degree. The initial efficiency gains of a quickly built app were over. Now we would be paying exponentially more in talent to scale.

We tried to find the most successful consumer Bubble app. We thought we could follow in their footsteps and scale to as least as large as they were. Unfortunately, *we* were the most successful of anyone we found. While you may think this was good for our egos, it was actually terrifying — we realized we couldn't scale on Bubble as far as we thought. Now we're experiencing an extreme form of *community limitations*.

Despite the challenges of dealing with these no-code limitations, we still scaled on Bubble 50% per month for another three months. To recap:

- Feature limitations: mainly building scheduling features and Google OAuth
- Infrastructure limitations: mainly with database queries taking too long
- Community limitations: limited experience with scalable web app development, small community, few professionals to freelance, essentially none who were candidates for full time, high prices for those who did freelance.



Moving to a codebase

Since Bubble was and still is the pinnacle of a no-code platform, and we were maxed out, now our only option was a code-based platform. At first we thought we could just hire someone to rebuild the platform in code. We tried that and realized we needed a “technical co-founder” instead of an employee: someone who could act as our CTO, going all in for the next few years while we built the company.

We spent the next 3 months hiring freelancers, posting jobs for a cofounder position, and networking around Nashville before finding the right person. He said one of the biggest reasons he wanted to join the company was the traction we’d already demonstrated.

Over the next 2 and a half months, our new CTO rebuilt the app in Rails and we were ready to make the switch. This was the core scope of the transfer:

- Data import: CTO instructed how he needed the data setup for import. I spent a day or so organizing that in spreadsheets.
- 301 redirects: Routes needed to change, so I crawled our website with Screaming Frog to get a list of the URLs and created a spreadsheet mapping them over. Here's a template.
- 3rd party tool connections: I reconfigured Zapier workflows to be triggered by a webhook. Our CTO programmed our app to send JSON data to Zapier at the appropriate workflow step in Rails.
- User email: Each of our users needed to do a password reset process to capture their account. My cofounder drafted an email to users that we planned to send at launch.

From here, our product was no longer bound by any of the *no-code limitations*. With our new team and code platform ready, we applied to YC that spring and secured funding for the summer 2017 batch.

Reflection and takeaways

Our tactic of hacking things together with the Zapier/Squarespace/Acuity stack was a great choice and proved our concept. We may have been better off moving straight to hiring a technical cofounder rather than rebuilding in Bubble. I think we were overly optimistic about no-code's potential as the core platform.

That said, we still believe in and employ no-code prolifically. We've built thousands of web pages using Webflow's CMS and a reverse proxy setup on top of our Rails app. One was the prototype for the Showcase product. It grew organic Google traffic 30% a week for 4 months until we moved it to Rails after it hit infrastructure and feature limitations.

For some pages we've built like the Sixty blog and our Insights product, they may never move off Webflow.

If Sixty started over, I would actually have Webflow power all our marketing pages. Bonsai and Alto are doing this and I'm also seeing many other Y Combinator startups build this way. It's a powerful setup — the marketing team can make changes to the site without burdening the engineering team.

CHAPTER VIII

The impact no-code will have on the world

The no-code movement, at its core, is about empowerment, and unlocking potential.

— Barrett Johnson, Product Marketing Manager

At its core, the no-code movement rests upon the fundamental belief that technology should enable and facilitate creation, not be a barrier to entry.

So, what impact will no-code have on the world? Put simply, the world of no-code is a world where ideas are not impeded by technological barriers — a world where anyone with an idea or a business they want to start will have the power to do it, without needing to learn code or find an engineer to make it a reality for them.

As has been the case in other industries, when the means of creating are made more accessible, the volume and scale of what's created skyrockets. In the world of software, the no-code movement is trying to do the same thing.

In any case, we're proud to be at the forefront of this new movement, and we can't wait to see what everyone builds.

CHAPTER IX

Learning resources

Connect with the no-code community and learn how to build, automate, and scale software without writing code.

Makerpad is a leading no-code community and learning center. Explore the best no-code tools, learn how to build and scale powerful applications, and hire no-code experts to help with your projects.

Webflow University is the go to place to learn the ins and outs of web design and development — without coding. Get comprehensive tutorials on designing and building dynamic websites, detailed featured documentations, and answers to frequently asked questions.

Nucode is a no-code community and forum where makers gather to learn and discuss various no-code projects. Learn how to build apps without any coding experience and engage with no-code experts around the world.

Nocode.tech is a curated directory of free resources and tools for non-technical creatives. Get discounts on no-code tools and learn the skills used by entrepreneurs, designers, and employees around the world.

Nocode HQ is a resource center for discovering and learning how to build with no-code tools. Learn how to build websites, mobile apps, chatbots, and automated workflows all without writing any code.

The Zapier Learning Center is an educational resource for learning how to automate everyday tasks without writing code. From project management to customer support, learn how to create automated workflows with Zapier.

Flowbase is a Webflow resource site for no-code beginners and experts. Get access to cloneable assets, templates, guides, and more.

Nocode Essentials.

Nocode Essentials is a central hub for discovering no-code resources. Easily find what you need to know for building, launching, and scaling your next project — without writing code.

CHAPTER X

No-code tool directory

From visual web development, to automation tools, no-code platforms enable teams to work faster and lower production costs.

Web development

No-code development platforms designed to help teams build beautiful and powerful websites.

Webflow is the way to design, build, and launch powerful websites — without writing code. Build professional websites, launch with Webflow's word-class hosting, or export your code and experience the power of HTML, CSS, and JavaScript in a 100% visual canvas. If you can design it, you can build it, with Webflow.

Carrd is a free platform for building simple, fully responsive one-page websites. Whether it's a portfolio or landing page, Carrd lets you build it beautifully and easily. Once your website is ready, publish it to the web with Carrd Pro.

Shopify allows you to create and manage ecommerce websites. Build and host your store, fulfil orders, and use Shopify's plugin marketplace to market your products — all in one place. From front-end to back-end, Shopify has you covered.

App development

No-code development platforms designed to help teams build powerful web and mobile apps.

Thunkable is a drag and drop mobile app builder that lets you create iOS and Android apps without coding. Apps built on Thunkable can work across all devices and can be directly published in the iOS App Store or Google Play Store.

Bubble is a visual programming language and application platform that lets you create interactive, multi-user applications for desktop and mobile browsers. Build out logic and manage a database without writing a single line of code.

Voiceflow allows you to design, prototype, and build voice apps. Create and publish Alexa Skills and Google Actions without writing code.

Glide lets you turn spreadsheets into apps, without code. Create easy-to-use apps in just 5 minutes from a Google Sheet.

AR/VR development

No-code development platforms designed for helping teams create powerful augmented and virtual reality apps.

Scapic lets you create augmented reality shopping experiences. Create 360 degree product visualizations and convert more customers through digital product engagement, all without writing code.

Automation

No-code automation tools designed to help teams automate workflows and connect their favorite apps together.

Zapier moves information between your web apps automatically. Connect over 2,000 apps together to build and automate workflows — without writing a single line of code.

Integromat allows you to connect your favorite apps and services together. Automate manual processes, transfer and transform data between applications, and save time on repetitive tasks.

Parabola lets you automate routine tasks. Input data into Parabola and visually create logic flows based on desired outputs. Process data, build CRM workflows, automate email marketing, and much more with Parabola.

Analytics

Analytic tools designed to help teams collect and understand data — from website analytics to user behavior.

Segment is a customer data platform that helps you collect and analyze your customer data. Use Segment as a central hub

for all your website and product data. Then, send that data to other platforms like Mixpanel, Google Analytics, or HubSpot to receive further insights. Empower your marketing, product, and engineering teams with clean data — collected with Segment.

Mixpanel is a product and user behavioral analytics platform that allows teams to get valuable customer insights. Import data from platforms like Segment, and analyze, measure, and improve your customer experience through data-driven insights.

Amplitude is a product analytics software for web and mobile applications. From SaaS to Fintech, Amplitude allows businesses to make data-driven decisions, based on user behavior. Build great product experience and retain users — with Amplitude.

Spreadsheets & databases

No-code spreadsheets & databases designed to help teams collect and store data.

Airtable works like a spreadsheet similar to Google Sheets, but gives you the power of a database to organize anything. From project management to a full CRM, Airtable allows you to build workspaces that can act as dynamic databases — without writing code.

dashdash is a spreadsheet that lets you build powerful sales and marketing tools. Then, you can publish them as web apps, without code. Create functions to automate repetitive tasks, integrate SaaS tools to collect data, and transform spreadsheets into web apps — all in one click.

Document editors

Document editors designed for helping teams keep track of databases, knowledge management, and project tasks.

Coda docs allow teams, big and small, to create documents and turn them into apps. From project management, inventory management, and remote collaboration, Coda brings everything into one place. Give a home for all your docs, spreadsheets, data, and workflow apps — without ping-ponging between applications.

Notion is an all-in-one workspace for knowledge management, tasks, notes, and databases. It acts as a central hub for teams to write, plan, and collaborate. With Notion you can replace tools like Google Docs, Evernote, Asana, Trello, and Google Sheets.

CRM

Customer relationship management tools designed for helping teams keep track of customers every step of the way.

HubSpot is a customer relationship management platform for marketing and sales teams. Collect, track, and organize inbound leads, and guide customers through every step of the customer journey.

Copper is a customer relationship management platform built for Google. Organize contacts, track leads at every stage of the sales cycle, and automate recurring tasks — all in G Suite.

Forms

Form builders designed for helping teams collect data and build personalized customer experiences.

Typeform lets you build interactive experience for surveys and lead generation forms. Attract more responses and increase response rates by turning data collection into an experience — without writing code.

JotForm is an online form builder designed for generating leads, distributing surveys, collecting payments, and more.

Email

No-code email tools designed for email marketing and helping creators monetize their content online.

Mailchimp is an all-in-one marketing platform, with a focus on email marketing. Grow your audience with signup forms and integrate them directly onto your website. Craft beautiful emails, create landing pages, and collect and manage leads — all in Mailchimp.

ConvertKit is an email marketing platform for online creators. Get the tools and automation you need to collect and nurture leads for your business or blog. Increase conversion rates with automated emails and get insights into what stage your subscribers are in the custom journey.

Substack allows online creators to easily create subscription-based email newsletters. Create your own publication, start your own newsletter, and get paid for writing about what you love.

Customer support

Customer support tools designed for engaging in conversations with existing and potential customers.

Intercom is a customer messaging platform that allows you to connect with your customers. It helps drive loyalty and growth at every stage of your customer lifecycle. Integrate chatbots on your website to support sales, marketing, and customer support teams — all with Intercom.

Payments

Online payment processors designed for sending and receiving payments on any website or app.

Stripe is an online payment processing tool for internet businesses. Easily connect Stripe to your website with its suite of payment APIs, designed for any size business.

PayPal is an online payments system that allows you to accept and send payments. Use PayPal to pay, send, or accept payments for almost any business.

User research

User research tools designed to help gain user insights to make data-driven decisions.

Optimal Workshop is a user research platform that helps you make decisions from user data. Transform insights into action and enhance your user research, UX design, or information architecture.

Prototyping

Prototyping tools designed for individuals and teams to go from ideation to final artwork.

InVision Studio is a screen design tool that allows teams to design, prototype, and animate — all in one place.

Figma is a collaborative interface design tool designed for rapid prototyping and gathering feedback. Create, test, and ship better designs — all in one place.

Sketch is a digital design toolkit designed for creating, prototyping, and collaborating on designs. From freelancers to large teams, Sketch helps designers go from idea to final artwork.

Adobe XD is a UI/UX design and collaboration tool created for teams of all sizes. Create designs and prototypes for websites, mobile apps, voice interfaces, games, and more.

Framer is a lightning fast interactive design tool made to bring creative ideas of life. Create responsive layouts and design functional prototypes with Framer.

webflow

Break the code barrier