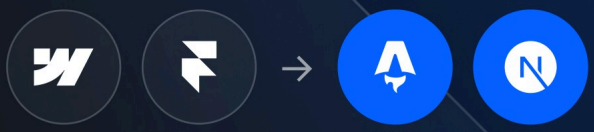
DESIGNME

FREE GUIDE · NOVEMBER 2026



Leave Webflow or Framer without losing a URL

The playbook from our own move to Astro: six days, a homepage that went from 10.33MB to 0.52MB, and every page that ranked still on its address.



Adrian Kuleszo
Co-founder, DesignMe

designme.agency

INSIDE

Nine chapters

01 **Our own move, in numbers**

What changed when we moved designme.agency from Framer to Astro, and what didn’t.

02 **Should you move at all?**

The honest test: when to stay on Webflow or Framer, and the signs it’s time to go.

03 **Astro or Next.js**

Which framework fits a marketing site, which fits a site that’s part of your product.

04 **What moves, and what breaks**

Content, CMS, images, forms, effects and redirects: what comes with you and what gets rebuilt.

05 **The redirect map**

Where rankings get lost, how to keep them, the 116-row map from our own move and the file it becomes.

06 **Choosing the CMS**

Markdown, Keystatic or Sanity: who edits what, and why you decide it before the build.

07 **The build, in order**

Six steps, from the URL inventory to a staging site checked page by page.

08 **Launch week, and after**

The switch-day plan, a way back, what to watch after, and the full checklist to print.

09 **Timeline and price**

How long a move really takes, what it costs with us, and what it costs to run.

SOUND FAMILIAR?

Every edit waits for the editor

A new team member or a pricing tweak sits in a queue, because only one person is comfortable in the canvas.

Pages are heavy, and you can’t fix it

The builder’s runtime ships on every page. Our homepage downloaded 5.59MB of JavaScript before anyone scrolled.

You pay more for a site you don’t own

Plans, editor seats and locales add up every year, and the site still lives on someone else’s platform.

You’re scared of losing rankings

Search brings in pipeline, and a move done badly can undo years of it. So the site stays where it is.

CHAPTER 01

Our own move, in numbers

We ran designme.agency on Framer for three years. In October 2026 we moved it to Astro: first commit on 7 October, live on 13 October.

Six days.

> Six days was our own site, our own team and no sign-off rounds. Don’t plan a client site on that number. Chapter 9 has the real timeline.

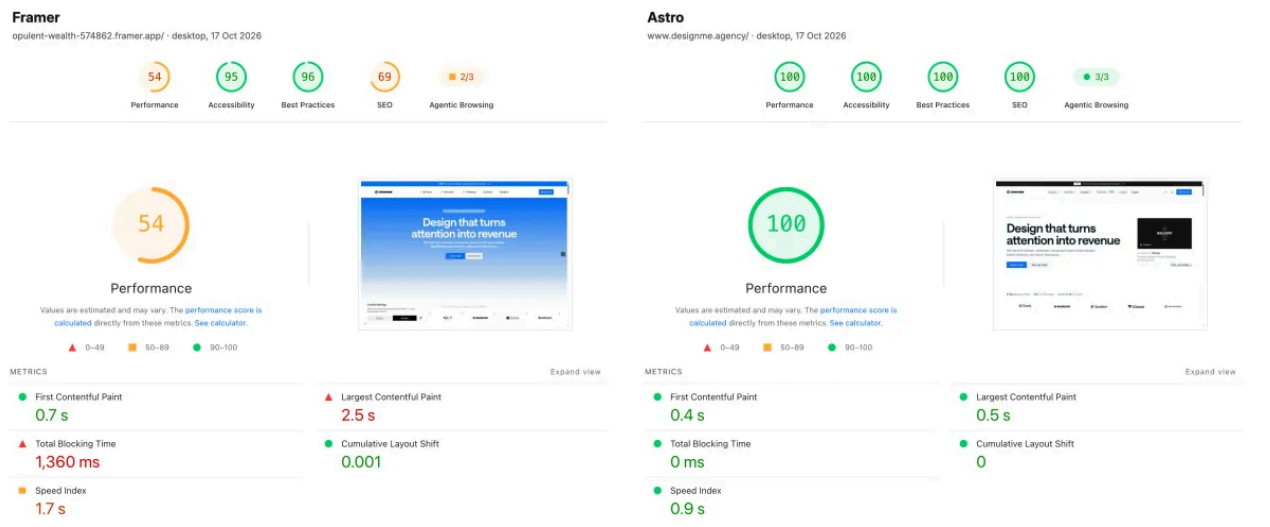
91 to 99
mobile PageSpeed Performance on Astro, from 33 to 71 on Framer

$0
a month to host on Cloudflare, from about $150 on Framer

15 min
at most for a typical page edit with Claude, from 3 to 5 hours

91
changes merged in the six days after the switch

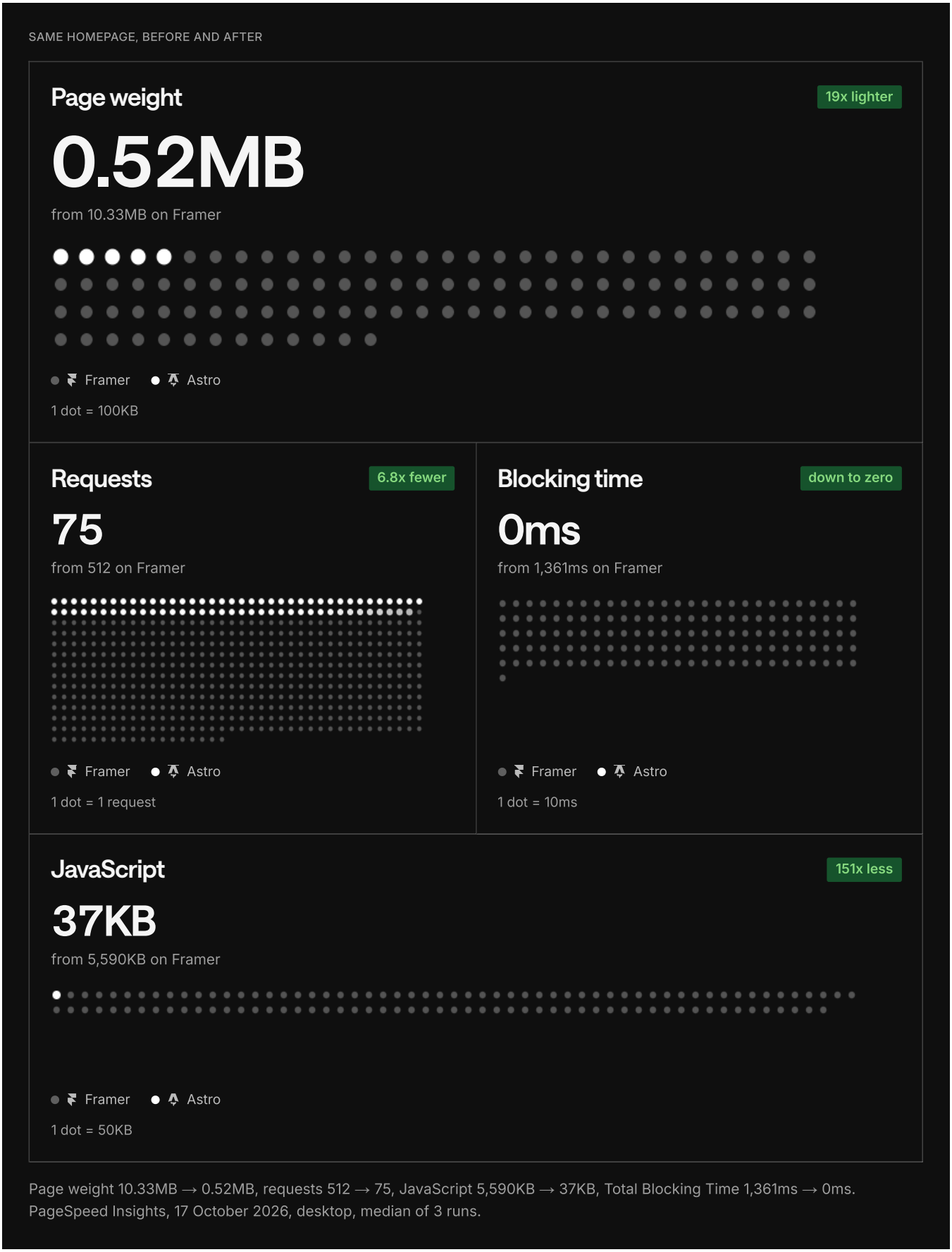
Our homepage, before and after Astro



Framer vs Astro, PageSpeed desktop

designme.agency

Here’s what changed on the homepage, measured with PageSpeed Insights on 17 October 2026 (desktop, the median of 3 runs on each site). One dot is one unit: grey is Framer, white is Astro.



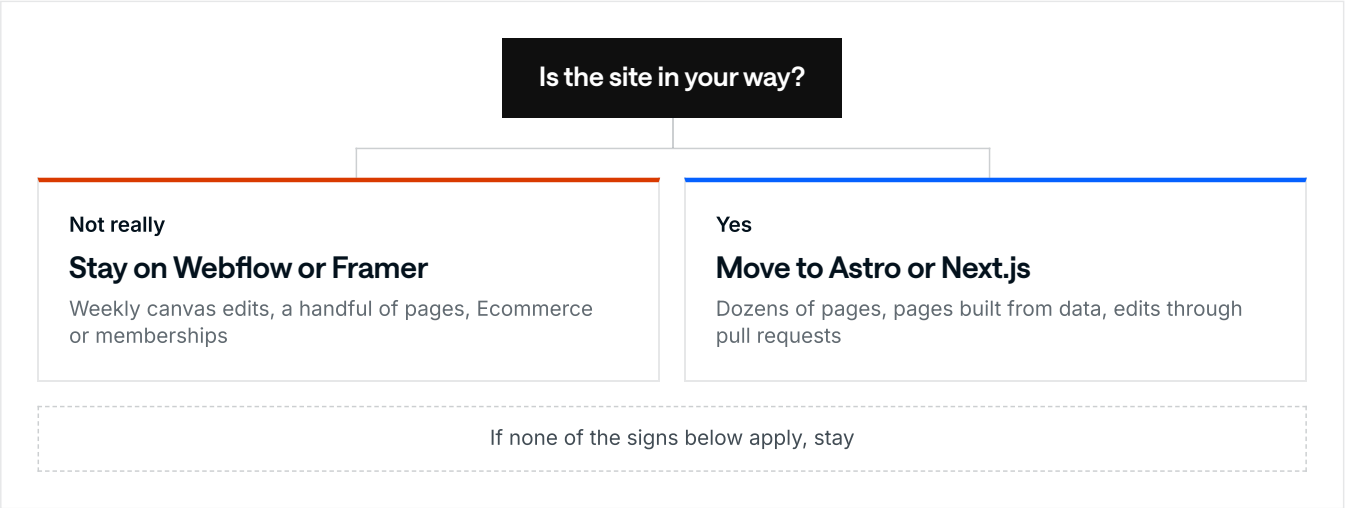
Not everything improved, and you should know that before you move: the homepage's mobile LCP stayed at 2.9 seconds on both sites. A new framework doesn't fix the largest image on the page for you.

And the part I care about most: most of those 91 changes started as one sentence to Claude.

CHAPTER 02

Should you move at all?

Not always. Webflow and Framer are great tools, and we still build on both for clients. Here's the test we use.



Stay where you are if

- Your marketing team builds new page layouts on the canvas every week, and nobody on the team works in code or with coding agents.
- The site is a handful of pages and the plan bill is the only thing bothering you. The move costs more than years of the plan.
- You rely on Webflow Ecommerce, memberships or localization.
- You need visual editing with no developer involved, ever.

Move if

- The site has grown into dozens of pages and the design has drifted. (Ours had 50+ colors and 30+ font sizes, added one section at a time.)
- You want pages built from data: comparisons, integrations, locations, docs. Same template, different content, and that's exactly what code is good at.
- You want your team (and Claude, Cursor or Codex) to change the site through reviewed pull requests, without waiting on whoever knows the editor.
- You want to own the site: a repository you control, no per-site plan, no editor seats, hosting wherever you like.
- Page weight matters to you, and the builder's runtime is something you can't remove.

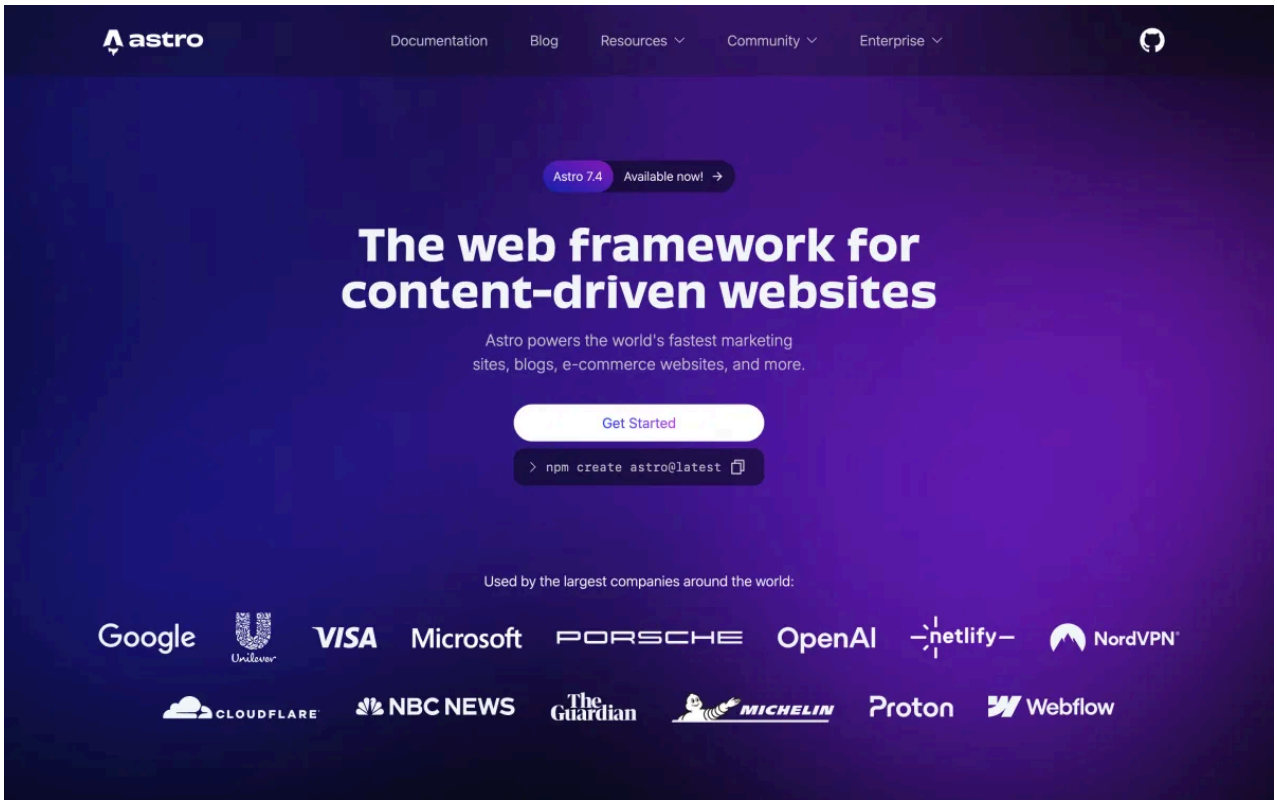
If none of these apply, stay. Seriously.

CHAPTER 03

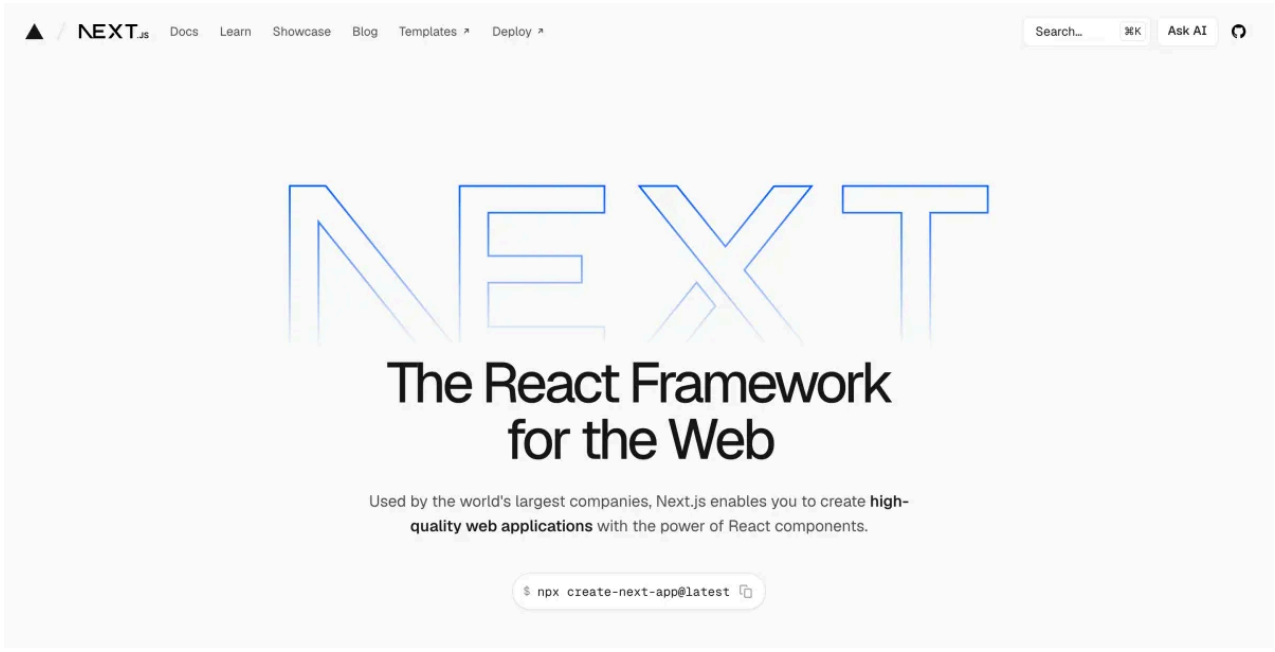
Astro or Next.js

Both are good. They're built for different jobs.

The two frameworks



astro.build — Astro



nextjs.org — Next.js

Pick Astro when most pages are the same for every visitor: a marketing site, a blog, docs, programmatic pages. Astro renders pages to HTML and ships no JavaScript unless a component asks for it. That's why we picked it for our own site.

Pick Next.js when the site behaves like part of your product:

- It shares components, a design system or authentication with a React app.
- There are logged-in areas, account pages or gated content.
- Pages are personalised per visitor, or generated per request from a database.
- Your engineers already ship Next.js every day and will own the site.

Astro and Next.js for a marketing site

	Astro	Next.js
How pages build	HTML and CSS, JavaScript only in islands	React end to end, static and dynamic parts mixed
Publishing from a CMS	A rebuild that takes a minute or two	Changes can appear without a rebuild
Hosting	Static files on any host	Fits Vercel best, runs elsewhere
Server code to patch	Very little on a static site	More on a server-rendered site

Some teams run both. Everything else in this guide works for either.

Running both

Astro

The main domain

Marketing, blog and docs

Next.js

The app subdomain

The product

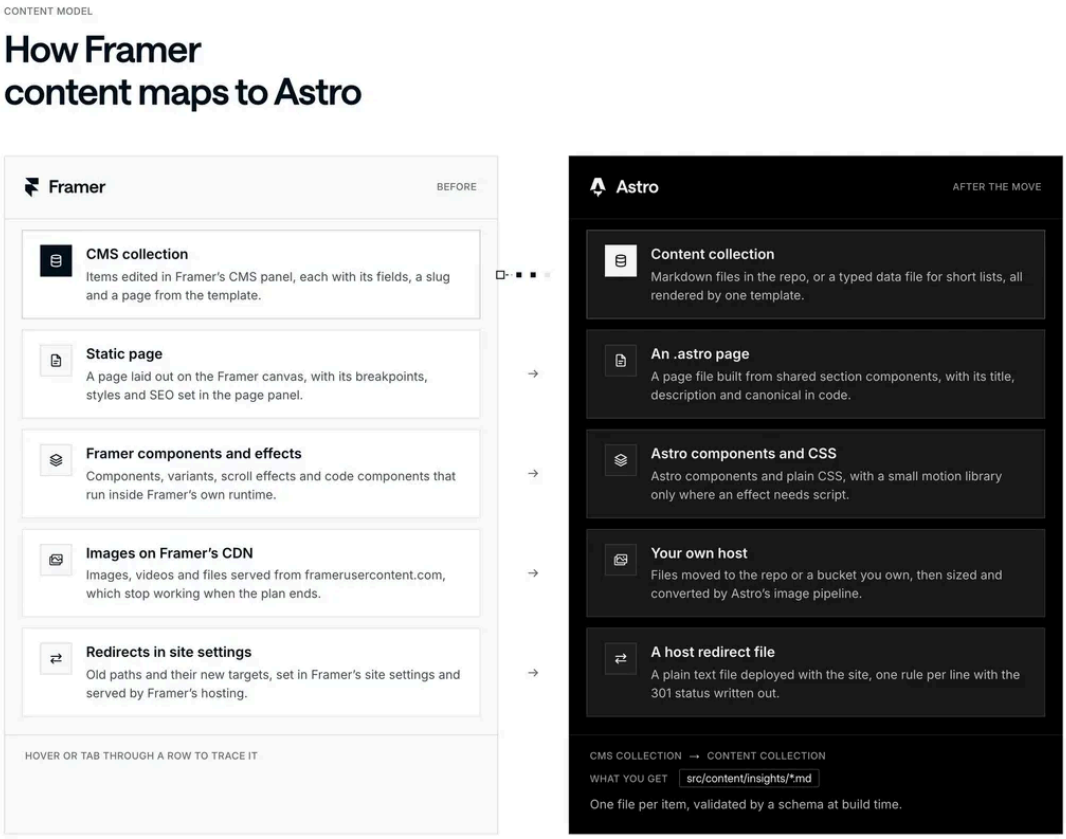
Both share design tokens, so visitors see one company

CHAPTER 04

What moves, and what breaks

Here’s the part nobody tells you upfront: neither platform gives you a site you can build on. Framer has no HTML export at all. Webflow’s code export gives you HTML and CSS, but CMS lists come out empty, forms lose their backend and only the primary locale is included.

So the move is a rebuild, not a conversion. The content comes with you. The layout doesn’t.



Content map designme.agency

From our own move: every part of a Framer site has a home in code. The layout is the only thing you rebuild.

PART OF THE SITE	FROM WEBFLOW	FROM FRAMER	ON ASTRO OR NEXT.JS
CMS content	Each collection exports to CSV (or comes out through the Data API), rich text as HTML	Each collection exports to CSV with Framer's CMS Export plugin	Markdown or typed data files, checked against a schema at build time
Page copy	Saved from the live pages	Saved from the live pages	Copied word for word from the saved HTML, nothing retyped from memory
Images and files	On Webflow's hosting, linked from the CSV	On framerusercontent.com	Downloaded and served from your own host
Pages and components	Rebuilt	Rebuilt	Rebuilt on a design system with tokens
Interactions and effects	Rebuilt	Rebuilt	CSS or a small motion library, shipped only where used
Forms	Lose their backend	Need a new home	A form service, your CRM or a serverless function
Redirects	Exported as CSV	Copied out of site settings	Real 301s in the host's redirect file

What breaks quietly

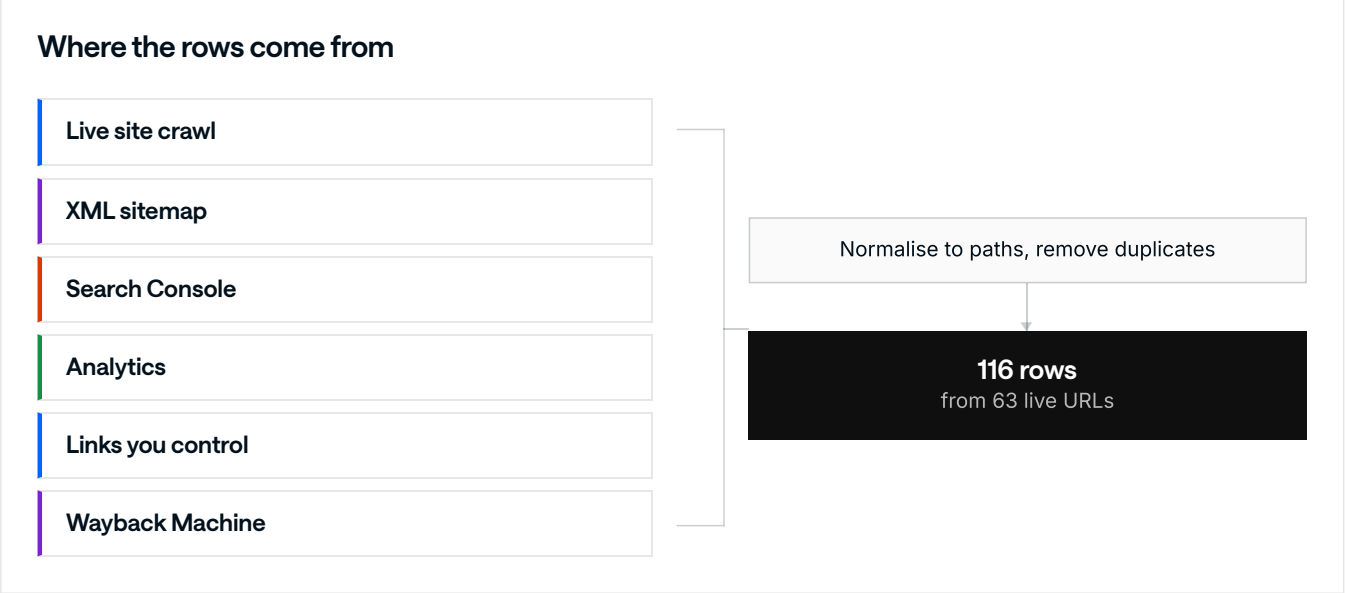
- **Images on the old CDN.** They still load on launch day, then stop working when you cancel the plan. Search the built site for Webflow and framerusercontent.com URLs before launch. We moved 1,378 files off Framer's CDN.
- **Pages missing from the sitemap.** Our own Framer sitemap was missing a page the navigation linked to. Crawl the site as well as reading the sitemap.
- **Trailing slashes.** Framer serves URLs without one. The new config has to match, or every URL quietly changes.
- **Redirects that aren't 301s.** On a static Astro build, redirects in the Astro config become meta refresh pages, and Cloudflare Pages uses 302 unless you write 301.
- **Editing.** The marketer who published on a Friday afternoon can't any more, unless you decided who edits what before the build (chapter 6).

CHAPTER 05

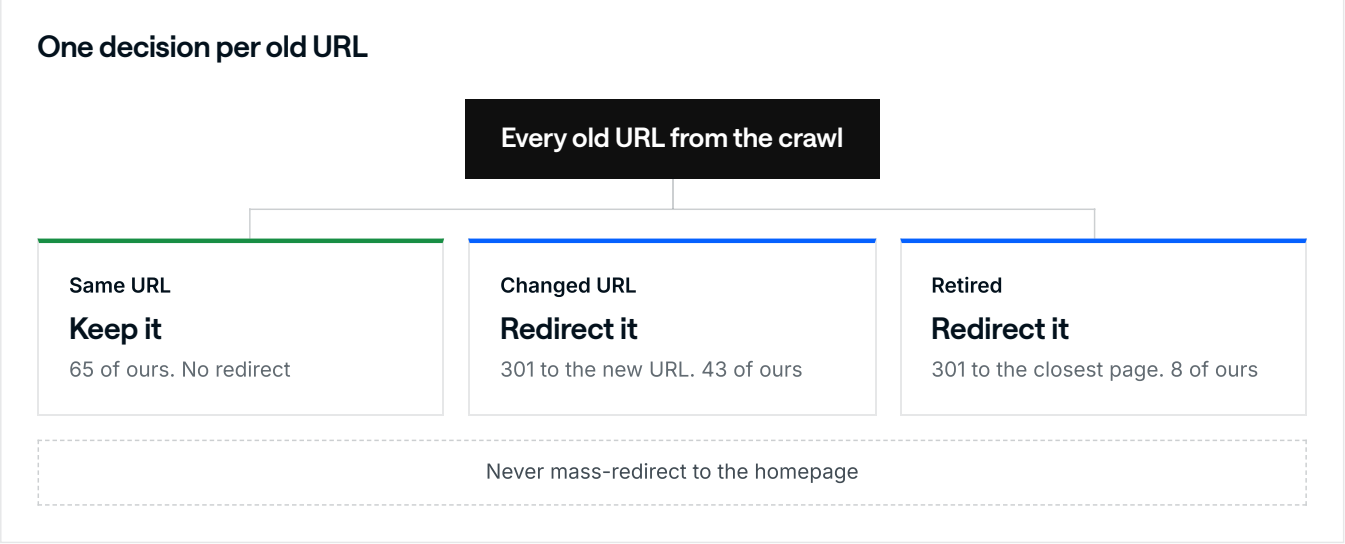
The redirect map

The redirect map is the part most migrations get wrong, so start there, before anyone designs anything.

Our own site had 63 live pages. The map had 116 rows, because every source of old URLs turns up pages the others miss:



Then every row gets one of three outcomes:



How to build it

01 **Collect every old URL**
From a crawl, the sitemap, Search Console, analytics, campaign links and the Wayback Machine.

02 **Keep paths wherever you can**
Astro and Next.js let you choose every route, so there's rarely a reason to change one.

03 **One hop to the closest page**
No chains, and never dump retired pages on the homepage.

04 **Write the host's format**
And check its default status code: Cloudflare Pages uses 302 unless you write 301.

05 **Test, then watch**
Every row before launch, 404s daily after it. Keep the redirects for at least a year.

The first week of 404 reports turned up a few more old links, and our map is now 124 rows. Every page that earned links or rankings still has a working address.

What the file looks like

On Cloudflare Pages, where designme.agency runs, the map is a plain text file called `_redirects` in the build: old path, new path, status code, one row per line. Four rows from ours, the last one a retired page sent to the closest page:

public/_redirects

```
# old path          new path                status
/call               /intro                  301
/about              /about-us               301
/privacy-policy     /legal/privacy-policy   301
/projects/recruitu  /projects               301
```

Write the `301`. Leave it off and Cloudflare Pages sends a 302, which tells Google the move is temporary.

On Next.js the same rows go in `next.config.js`. `permanent: true` sends a 308, which Google treats like a 301:

next.config.js

```
module.exports = {
  async redirects() {
    return [
      { source: '/call', destination: '/intro', permanent: true },
      { source: '/about', destination: '/about-us', permanent: true },
    ];
  },
};
```

Keep the map itself in a spreadsheet (old URL, new URL, outcome) and generate the file from it. Ours lives in the repository as a CSV, and a script writes `_redirects` from it, so nobody edits the file by hand.

> Want a head start? The free migration planner lists every URL on your site and drafts the redirect map for you.

Rather hand the redirect map over?

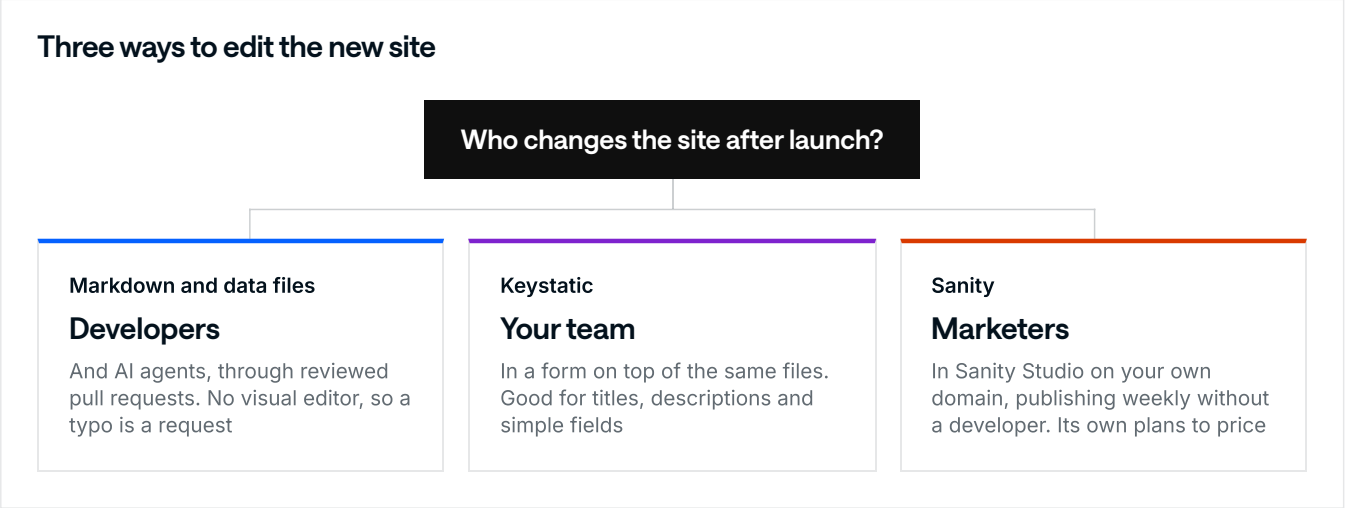
Bring your URL to a 30-minute call. You'll leave knowing which stack fits, how long the move takes and what it costs, redirect map included.

Book a 30-minute call

CHAPTER 06

Choosing the CMS

Decide who edits what before the build, not after. It changes the build.



Here’s what we run: insights are Markdown files in the repository, the rest is typed data, and Keystatic covers page titles and descriptions. Everything else changes by asking Claude, and nothing goes live until a person approves the preview.

If your marketers want to publish a post on their own on a Friday afternoon, give them a CMS. Both Astro and Next.js work well with Sanity. (On a static Astro site, a publish triggers a rebuild that takes a minute or two.)

CHAPTER 07

The build, in order

This is the order we used on designme.agency, and the order we use for clients.

From inventory to staging

01 Before any code

Inventory every URL

Give each one a route in the new project before a line of code.

02 Before any code

Save the source

The raw HTML of every page, every CMS collection as CSV, every asset mapped from its old URL to a local file.

03 Build

Design system first

The builder’s values become tokens (type, color, spacing, light and dark), then shared sections. Pages are assembled from those.

04 Build

Parity, then improvements

The first version matches the live site page for page: same copy, same SEO. Changes come after sign-off.

05 Before launch

SEO on every page

Titles, descriptions, canonicals, OG images and structured data, plus a sitemap with only indexable pages.

06 Before launch

Staging, side by side

The full site on a private address, compared with the old one page by page.

Parity first is the rule people want to skip. Don’t. If traffic moves after launch, you’ll know it wasn’t the copy.

CHAPTER 08

Launch week, and after

Launch early in the week, so someone is around when Google starts re-crawling.

Around the switch

Before DNS moves

- **Redirects tested**
  Every row of the map, on staging.
- **No staging noindex**
  Checked on every template.
- **No old CDN links**
  In images, videos, OG tags or CSS.

Launch day

- **A way back**
  Ours was one DNS change, with a five-minute path back to Framer.
- **Sitemap submitted**
  The new one, in Search Console.
- **Forms and analytics**
  Tested on the live domain, with cookie consent set up.

The first weeks

- **404s and indexing**
  Watched daily until the numbers settle.
- **The old plan kept**
  Until every asset is off it.

The launch checklist

Every step in this guide as one list. Print it, or copy it into your project tool.

BEFORE THE BUILD

- [ ] Every old URL collected: crawl, sitemap, Search Console, analytics, campaign links, Wayback Machine
- [ ] Every URL has an outcome: keep it, redirect it, or redirect it to the closest page
- [ ] The raw HTML, every CMS collection as CSV and every asset saved
- [ ] Who edits what after launch decided, and the CMS picked

BEFORE DNS MOVES

- [ ] Every page matches the live site: same copy, same SEO
- [ ] Titles, descriptions, canonicals, OG images and structured data on every page
- [ ] A sitemap with only indexable pages
- [ ] Every redirect row tested on staging: one hop, 301, never the homepage
- [ ] No staging noindex left on any template
- [ ] No old CDN links in images, videos, OG tags or CSS

LAUNCH DAY, EARLY IN THE WEEK

- [ ] A way back ready, such as one DNS change
- [ ] The new sitemap submitted in Search Console
- [ ] Forms and analytics tested on the live domain, cookie consent set up

THE FIRST WEEKS

- [ ] 404s and indexing checked daily until the numbers settle
- [ ] New 404s added to the redirect map
- [ ] The old plan kept until every asset is off it
- [ ] Redirects kept for at least a year

CHAPTER 09

Timeline and price

Our own move took six days. That was our site, our team, no sign-off rounds and Claude doing a lot of the typing. For a client site, we plan 4 to 5 weeks for most sites, from the URL inventory to launch. Heavy integrations, lots of custom effects or a big CMS take longer.

4 to 5 weeks

for most sites, from the URL inventory to launch

$10,000

the starting fixed price for a landing page sized site

$15,000

the starting fixed price for a full site or a headless CMS build

You get the scope and the price on a call, before anything starts, and the platform you pick doesn't change it.

Running costs usually drop. A mostly static Astro site costs little or nothing to host for most marketing sites, and there are no editor seats for the site itself. A CMS like Sanity has its own plans, so price it for your team.

Want a number for your site? Book a 30-minute call and bring your URL. You'll leave knowing which stack fits, roughly how long it takes and what it costs. And if the honest answer is "stay on Webflow" (or Framer), I'll tell you that too.

RATHER TALK IT THROUGH?

Want a plan for your own site?

Bring your URL to a 30-minute call with Adrian. You'll leave knowing which stack fits, how long the move takes and what it costs.

Book a 30-minute call at designme.agency/intro

01

Which stack fits

Astro, Next.js, or staying where you are. We'll say so if it's the last one.

02

How long it takes

A timeline for your page count, CMS and integrations.

03

What it costs

A fixed price from $10,000, confirmed before anything starts.

DESIGNME

DesignMe designs and builds websites for funded startups and B2B teams, in Framer, Webflow, Astro and Next.js.