LIZZYLAND

Code Your Own Site

The honest, no-nonsense start for building your own website — no coding background required, no agency invoice either.

lizzyland.xyz

A free resource. This is the honest overview, not the whole course — if you'd rather it built for you, that's on the Wealth door too.

· First, The Reframe ·

So You Want To Build Your Own Website?

It's easier than you think.

It helps if you already have some sense of how websites work — the rough idea that your browser is asking a computer somewhere else for information, and that computer is sending it back. Understanding that makes it much easier to troubleshoot when something goes wrong. But if you have zero experience, it's still absolutely possible for you. Everything below assumes you're starting from nothing.

This isn't the whole course — it's the honest overview. Enough to actually understand what you're doing and give it a proper go yourself. If you get partway through and decide you'd rather it just got built, that's what the paid options on the Wealth door are for. No hard feelings either way.

WORTH KNOWING

This whole document is really three jobs: getting your ingredients together, planning what you're building before you touch a keyboard, and then actually building it. Skipping the planning stage is the single most common reason people get stuck and give up halfway.

STEP 1

Gather Your Ingredients

A computer with internet access

Preferably an unlimited connection — this is a data-heavy exercise, with a fair bit of uploading and downloading as you go.

A domain name

This is your website's address — think of it as a P.O. Box for the internet, so people (and browsers) know exactly where to find you. Namecheap is a solid, affordable place to register one. Get yours here.

Somewhere for your code to live — GitHub

GitHub isn't hosting. It's where your website's code is stored and backed up, version by version. It's free, works entirely in your browser, and needs no download. Think of it as the filing cabinet, not the shopfront.

Somewhere to actually host your site — Railway

This is what takes the code sitting in GitHub and turns it into a real, live website the public can visit. Railway's free "Hobby" tier comfortably covers one small site to start.

Your graphics

A good-quality logo, a handful of themed banner images, and real photos of your product or service. Low-resolution or stretched images are one of the fastest ways to make a site look amateur — get these right before you start.

A sense of "the look"

Spend twenty minutes browsing sites in your space and bookmark the ones whose style actually appeals to you. You don't need to know why yet — just collect examples.

Your content

Your About page, your address and contact details, and the products or services you're offering with proper descriptions. Have this written before you start building — trying to write content and build a site at the same time is how projects stall.

There may be a thing or two above I haven't thought of for your specific site — that's normal, not a sign you're missing something critical.

ONE HABIT THAT SAVES YOU HOURS

Name your image files properly before you upload anything anywhere — the_sauna_hero.png, not dkaeyr3465tq25kgc.png. A camera or phone will hand you the second kind by default. Rename as you go, not after — future you will be very grateful.

STEP 2

Plan It Before You Build It

Most basic sites are three pages: a landing page, a contact page, and an about page. Everything else hangs off that.

Your Landing Page Has a Job — Know What It Is

This is the first thing anyone sees, so it needs to match what you're actually selling:

- Mostly a shop? Consider skipping the traditional hero-and-mission-statement opening and put your products front and centre instead — people came to buy.
- A blog? This is your chance to introduce yourself properly, with links straight to your best writing.
- Selling a service? Treat it like a reception area. People here are often in a hurry — make the path to booking or contacting you obvious within seconds.

About and Contact — One Page or Two?

If you don't have much to say, combine them into one page. If you've got a longer story to tell, or several contact points (multiple offices, departments, whatever applies), split them. Don't force a short story to fill a page it doesn't need, and don't cram a long one into a corner.

Think of Your Site as a House

Every anything-else beyond the three main pages needs to sit further down the hierarchy, reachable by a logical path — not floating on its own. Rooms should lead to the next logical room. You don't put a bathroom off the kitchen, and you don't build a passage that leads nowhere.

```
Home
  |-- About
  |-- Contact
  |-- Shop
  |   |-- Product Category A
  |   '-- Product Category B
  '-- Blog
      '-- Individual Post
```

Get a sheet of paper and sketch your own version of this before you touch a keyboard. This is called a sitemap — worth knowing the term, you'll see it again if you go looking for more on this. Do this properly now and you won't find yourself restructuring the whole site halfway through building it. Start with the end in mind.

Write It All Down — Your Brief

Open a plain text document and detail out your site properly. This becomes your brief, and it matters more than it sounds like it should:

- Your domain name
- Your site structure (the sitemap you just sketched)
- Your content — every page, with headings, and notes on what image goes where
- The other sites you drew inspiration from, and what specifically you liked about each
- Your choice of fonts and colours

Keep Everything in One Folder

Create a folder on your computer called something obvious — "My Website" — and put everything in it: the brief, the logo, the banners, the product photos. Nothing scattered across your phone, your downloads folder, and three different apps. When it's time to hand files to Claude or upload them anywhere, you want to reach for one place.

Note This in Your Brief: Speed Matters

Add a line reminding yourself that your images need to be optimised — compressed and reasonably sized, not straight off a 20-megapixel camera. A slow-loading site loses visitors before they've seen anything, and it's much easier to plan for this now than to fix it later across every page.

> **WHY THIS STEP FEELS SLOW**
>
> It is slow — on purpose. Every minute spent here is a minute you don't spend rebuilding a page you described badly the first time. This is the only step where being thorough actually pays for itself later.

STEP 3

Making Your Site With Claude

This is the part that used to require years of learning to code. It doesn't anymore — you're about to direct someone who already knows how, in plain English.

Start a Project, not just a chat

On the left-hand side of claude.ai, create a Project — this is different from an ordinary chat, because it keeps your files and instructions together in one place that Claude will reference automatically. Upload your brief document and your logo into the Project's Files section.

Open a chat and ask

Inside that Project, open a new chat. Briefly explain what you're building and ask Claude to build the site to your brief's specifications — it will read everything sitting in the Project by itself, you don't need to re-explain it.

Let Claude write the code

Claude will produce the actual files for your site right there in the chat — for a small site this might be one or two files, for a fuller one, several. Ask it to bundle everything into a single zip file so you get it all in one download, rather than grabbing files one by one.

Get it onto GitHub

Unzip the download on your computer. Create a free GitHub account if you don't have one, click "New Repository," and use the "upload an existing file" option to drag your whole unzipped folder in. Remember — this stores and backs up your code. It does not put your site online by itself.

Deploy it on Railway

In Railway: New Project -> Deploy from GitHub repo -> select the repository you just created. Railway reads your code and builds the live site automatically from there.

Add a Volume if your site needs to remember anything

If your site has a database, stores uploaded images, or saves blog posts, add a Volume in Railway (Settings -> Volumes). Think of it as a small permanent hard drive for your app, so nothing gets wiped every time you update the site.

View it, and fix what's wrong

Once the build finishes — usually a couple of minutes — click the Railway-generated link to see it live. Anything you're unhappy with, go back to your Claude chat and describe exactly what's wrong. If something is actually broken, copy the exact error message from your browser and paste it in word for word — this is the single fastest way to get a correct fix. Always ask for the complete file back, not a snippet, so you're never uploading a half-finished patch.

Repeat until you're happy

Fix, re-upload the changed file(s) to GitHub, and Railway redeploys itself automatically within a minute or two. Keep going. This back-and-forth is completely normal — it is not a sign you're doing something wrong.

Connect your domain

In Railway, go to your project's Settings -> Domains -> Custom Domain, and add the one you registered in Step 1. Railway will give you a value to paste in — usually a CNAME record. Log into Namecheap, open Advanced DNS for your domain, and add that CNAME record there. It can take anywhere from a few minutes to 24 hours to fully connect — that delay is completely normal.

· Two Things Worth Knowing ·

Before You Go

Never Put a Password or Key Directly in Your Code

If your site ever needs a password, an API key, or anything else secret to function, it does not belong typed directly into your files — especially once those files are sitting in a public GitHub repository. Railway has an "Environment Variables" section built exactly for this. Ask Claude to use one when the moment comes, and it will know what to do with it.

The Padlock Takes Care of Itself

HTTPS — the little padlock next to your web address — is handled automatically by Railway once your domain is connected. There's no certificate to buy and no separate step to remember.

When You'd Rather It Just Got Built

This is the honest, no-nonsense version — enough to actually do it yourself. If you get partway through and would rather hand it over, that's exactly what The Foundation and The Full Monty on the Wealth door are for. Same stack, same standards, built for you instead of by you.

lizzyland.xyz/wealth