Using AI to Actually Learn to Code

Not a shortcut around learning — a genuinely fast way to do it, if you use it right.

The wrong way to use it

It's easy to ask an AI tool for a finished piece of code, paste it in, watch it work, and move on having learned nothing from the exchange. That failure mode is worth naming up front, because avoiding it is most of what this guide is about.

The way that actually teaches you something

- Paste in a file or snippet you don't fully understand and ask it to walk through it line by line.
- Ask “why” as often as “what” — why this approach and not another, why this particular line is even needed.
- Once an example clicks, ask for a second, different example of the same concept. A second example is often what actually makes an idea stick.
- Ask it to intentionally break something and explain the resulting error — seeing what a mistake looks like is as useful as seeing what works.
- Build something small yourself first, get stuck, and then bring in the specific thing you're stuck on — not the whole project at once.

A real example of the loop

Here's roughly what that looks like in practice, from fixing an actual bug on this site:

- Write a first attempt at something — say, a mobile menu button — and get it partway working.
- Hit a wall: the menu opens, but the page behind it is somehow still visible through it.
- Paste the relevant CSS and ask why a fixed-position element would let the page behind it show through.
- Get an explanation of the actual mechanism at fault, not just a fixed version of the code — in this case, a CSS property that quietly changes how fixed positioning gets calculated.
- Ask for a second scenario where that same mechanism would cause a different bug, to check the concept actually stuck rather than just the fix.

That loop — attempt, get stuck, ask why, verify you understood, apply it somewhere else — is the same loop behind most of the fixes across this entire site.

Verify, don't just trust

AI explanations are sometimes wrong, especially about very new tools or specific version behavior. Treat an explanation as a strong hypothesis, not a fact: test it against what's actually happening on your screen, and if it doesn't match, say so and ask again instead of assuming you must have misunderstood.

Where this fits with the other guides

The HTML/CSS, JavaScript, and Python guides in this section give you a base vocabulary to actually have a conversation about. The more of that foundation you have going in, the better questions you can ask — and the better answers you'll get back. Used this way, an AI tool stops being a search engine and starts acting like a very patient tutor who never gets tired of being asked “wait, why?”

> *This is genuinely how a lot of this site got built — sending over a file, having it broken down piece by piece, and asking for more examples until each part actually made sense, instead of just accepting whatever came back first.*
