◆ FREE GUIDE

AI ESSENTIALS · GUIDE NO. 13

BUILD A REAL TOOL WITHOUT CODE

Describe a small business tool in plain English and AI builds a working version you can actually use — a quote calculator, an intake form, a dashboard — with zero code and, if you want, a link you can share.

~9 MIN READ · FOR EVERYONE WHO COMMENTED "BUILD" · ZERO CODE REQUIRED

Hank Barker // AI Essentials // @hankhelpsai

THE SHORT VERSION

- → If you can **describe** a small tool in plain English, AI can build you a working version — one you can click, type into, and use — with no code.
- → Three tools do it: **Claude Artifacts** (builds it AND runs it live, with a share link), **Gemini Canvas** (builds it and can wire AI into the tool), and **ChatGPT** (writes great code, but you host it).
- → The no-fuss format is a **single HTML file** — one text file your browser opens as a working page. Double-click it, email it, or drop it on free hosting.
- → The loop: **describe** → **generate** → **use it** → **tell it what's wrong** → **ship.** Never put passwords or API keys in a tool like this.

THE REFRAME

For years, building even the smallest internal tool meant finding a developer. You'd write up what you wanted, wait for a slot on someone's roadmap, maybe pay for it, and get it back a few weeks later — if the request survived at all. Most didn't. The little tools that would've saved your team ten minutes a day never got built, because they were never worth a developer's week.

That math just changed. If you can describe a tool clearly in plain English, AI can build you a working version of it — one you can actually click, type into, and use — in about the time it takes to explain it. No code. No ticket. No waiting. You describe it, the AI writes it, and it runs right in front of you.

This isn't about replacing your engineering team. Real software — the stuff with logins, databases, and customer records — still deserves real developers, and we'll be clear about where that line sits. This is about the **long tail of small internal tools** that never made the cut: the pricing calculator your sales reps keep rebuilding by hand, the intake form you email around as a Word doc, the checklist that lives in one person's head. Those are exactly what this is good at.

Picture the everyday version of this. Your sales team quotes jobs by opening last month's spreadsheet, copying a row, and hoping they remembered the rush surcharge. Everyone does it slightly differently, and once a quarter someone under-quotes and eats the difference. The "right" fix is a proper internal app, which never gets prioritized. The new fix is a ten-minute conversation that ends with a clean calculator everyone uses the same way. That's the whole shift, in one boring, valuable example.

The shift is simple, but it's a big one. The bottleneck used to be building. Now the bottleneck is knowing what you want. Get clear on the tool and the tool basically builds itself.

The bottleneck used to be building the tool. Now it's just being able to describe it.

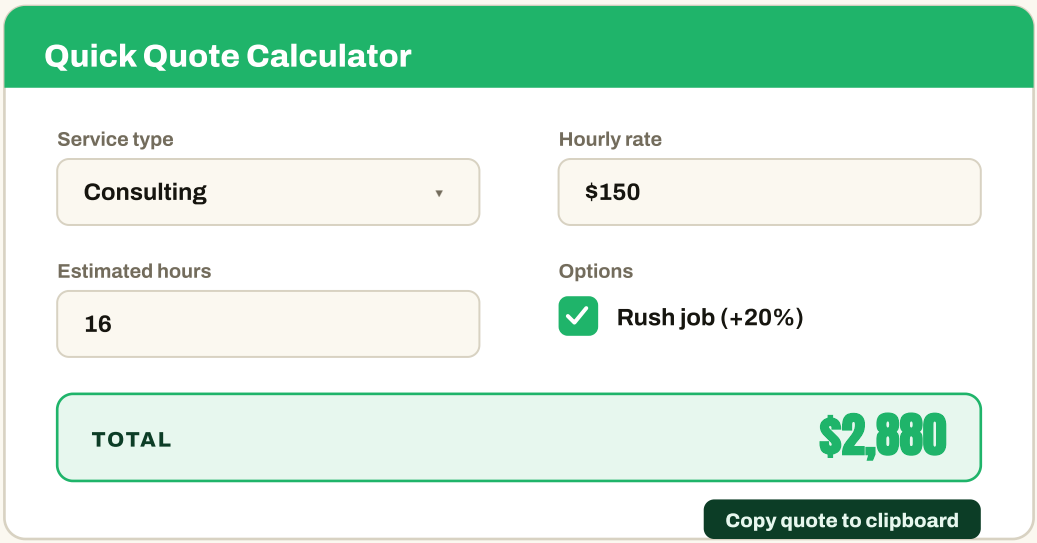
WHAT YOU CAN ACTUALLY BUILD

Keep it to small, self-contained tools that do one job well. Here's what's realistic — and genuinely useful — for someone running a business.

- **A quote or pricing calculator** — punch in a few numbers, get a clean total. Your team quotes the same way every time instead of eyeballing it in a spreadsheet.
- **A lead-intake form** — a tidy form that collects exactly the fields you need, so nothing's missing when a new lead lands on your desk.
- **A "cost of a meeting" calculator** — enter salaries and headcount, see what that standing meeting actually costs per year. Brutal, and great for making a point in a budget conversation.
- **A proposal or email generator** — fill in a few boxes, get a formatted proposal or a polished message you can copy straight out.
- **A simple dashboard** — paste in your numbers and see them laid out as clean cards or a small chart, instead of squinting at rows in Excel.
- **A checklist app** — a repeatable onboarding, closing, or handoff checklist your team ticks through the same way every time.

Notice what these have in common. Each does **one job**, serves **one team**, and doesn't need to store sensitive data or connect to your bank. That's the sweet spot. The moment a tool needs real accounts, saved records, or customer data, you've crossed into "hire a developer" territory — we'll come back to that. Here's what one of these actually looks like, drawn as a mockup:

A REAL TOOL, BUILT FROM ONE PROMPT



A quote calculator like this is a great first build — a few inputs, a live total, a copy button. Nothing here needs a developer.

THE THREE BUILD SURFACES, COMPARED HONESTLY

Three tools can do this today, and they are not the same. Picking the right one saves you a lot of fiddling, so here's the honest comparison — strengths and catches.

THREE WAYS TO BUILD — COMPARED HONESTLY

Claude Artifacts	Gemini Canvas	ChatGPT
Easiest end to end	AI you can build in	Great code, you host it
Builds it and runs it live	Builds and previews it	Writes clean code (Canvas)
One-click shareable link	Wire Gemini into the tool	No native run + host
No hosting to set up	Add AI features inside it	You arrange hosting
Refine it in plain English	Share or export it out	Or one self-contained file
BEST FOR	BEST FOR	BEST FOR
Shipping fast	AI-powered tools	Owning your hosting

All three can build. They differ on whether the tool runs live for you, whether you get a link, and whether you have to arrange hosting yourself.

1 · Claude Artifacts — the easiest end to end

You describe the tool in a normal Claude conversation, and Claude builds it in a panel next to the chat called an **Artifact** — and it runs right there. You can click the buttons, type in the fields, and use the thing immediately. Don't like something? Say so in plain English and it updates in place. When it's ready, you get a shareable link so a colleague can open and use it too. Describe, use, share — all in one window, no setup.

```
Build me a quote calculator: inputs for service type, hours, and rate, with a "rush job"
checkbox that adds 20%. Show the total live and add a "copy quote" button.
```

2 · Gemini Canvas — AI you can build into the tool

Google's version, called **Canvas**, also builds and previews apps live. Its standout is different: you can wire Gemini itself **into** the tool you're building. So the app isn't just buttons and math — it can have an AI feature working inside it. Think a tool that drafts the follow-up email, summarizes the intake notes, or rewrites a stiff quote in a warmer tone, all inside your own little app. If the thing you're imagining needs AI to be part of what it does, Canvas is the one to reach for.

```
Build a lead-intake form. After someone fills it in, use Gemini to write a short, friendly
summary of the lead that I can paste into our CRM.
```

3 · ChatGPT — great code, you handle hosting

ChatGPT is excellent at the actual writing of the code — its **Canvas** feature gives you a clean, editable code panel. The catch is there's no native "run it and host it as a live app" the way Artifacts has. So you've got two paths. One: ask it to make a **single self-contained HTML file** — then you just open that file and it works like a web page (we'll explain HTML in a second). Two: put the code somewhere that hosts it for you. For most non-technical people, the single-file route is the no-fuss one.

```
Build me a "cost of a meeting" calculator as a single self-contained HTML file — inputs for
number of people, average hourly cost, and meeting length. Nothing to install; it should
just open in a browser.
```

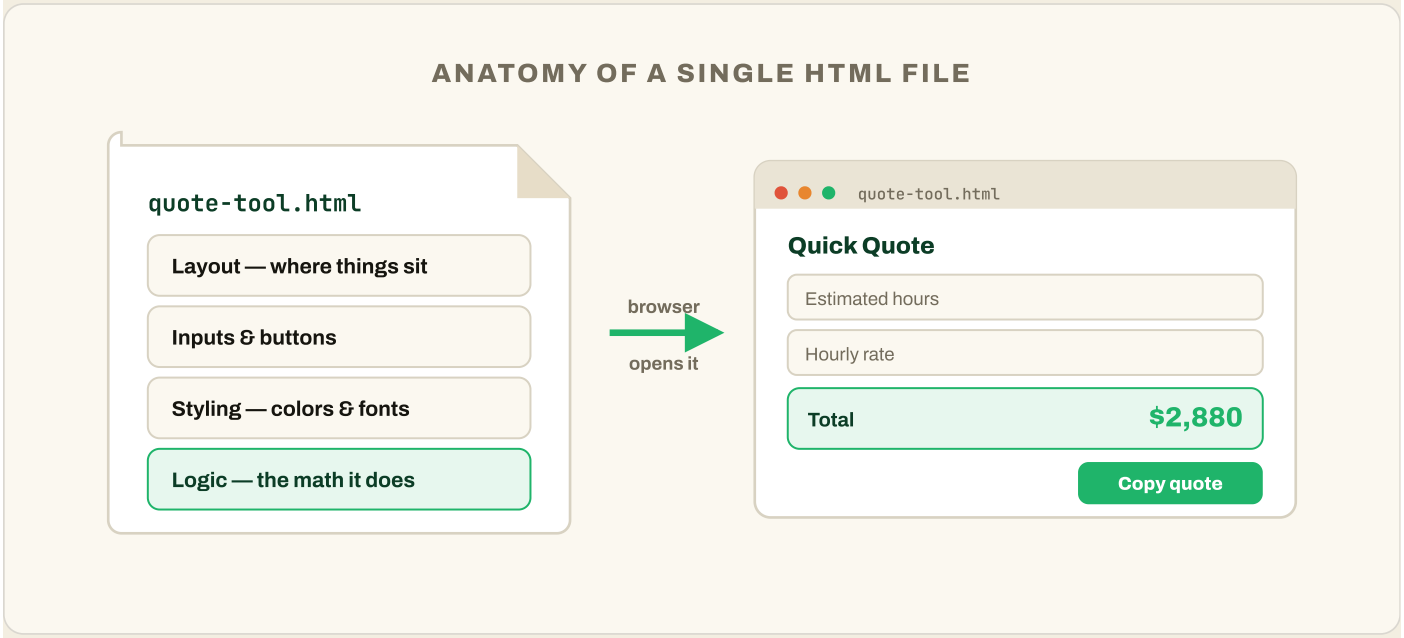
QUICK PICK

Want it fastest and shareable? **Artifacts.** Want AI features living inside the tool? **Gemini Canvas.** Already in ChatGPT and happy to open a file yourself? Ask for a **single self-contained HTML file** and you're done.

WHAT HTML ACTUALLY IS

That phrase — "single HTML file" — is the key to this whole thing working without a developer, so let's demystify it. No jargon.

HTML is just a **text file**. One file. Your web browser knows how to open it and turn it into a working page — the buttons, the input boxes, the layout, the colors, and the logic that does the math are all written inside that one file. When you build a tool this way, everything lives in a single document. There's no app to install and nothing to configure.



Layout, inputs, styling, and the logic all live in one file. The browser reads it and turns it into a page you can use.

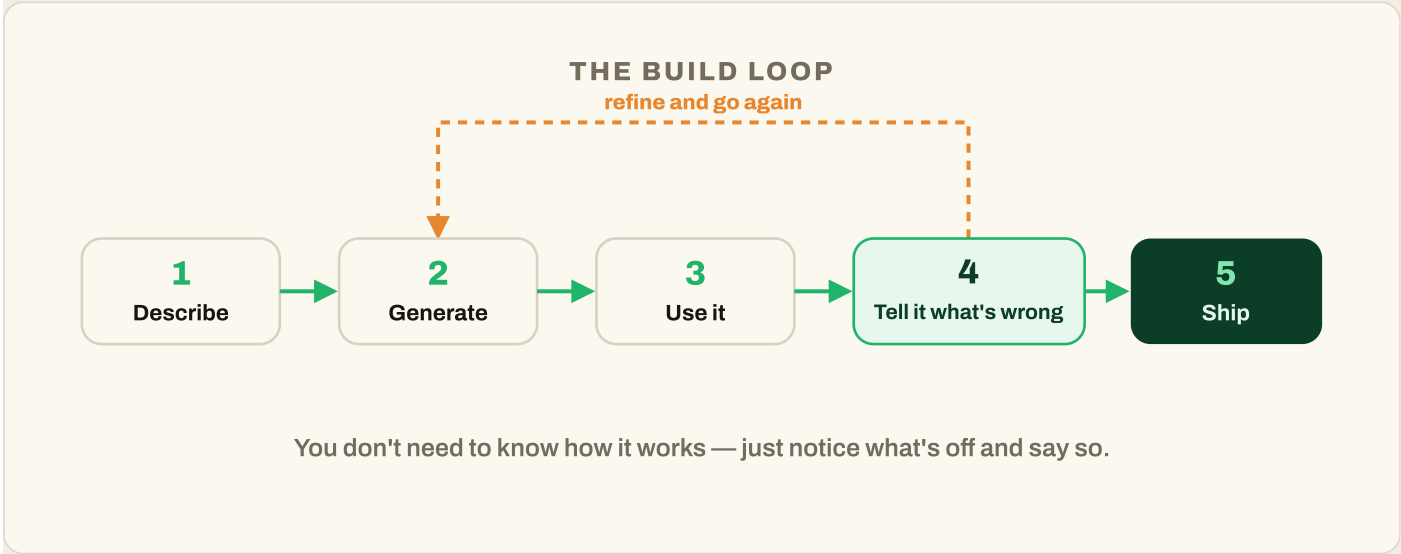
That's why single-file HTML is the no-fuss path. Save the file to your computer and **double-click it** — it opens in your browser and works like any website, except it lives on your machine. Want to share it? Email the file, and whoever opens it gets the working tool. Want a real web link? Drop that same file onto free hosting and you've got a URL. One file, three ways to use it — and you never touched a line of code.

ASK FOR IT BY NAME

When you use ChatGPT or any of them, say "make it a **single self-contained HTML file**." "Self-contained" means everything is baked into that one file — no extra pieces to wire up. That one phrase is the difference between a tool that just opens and a pile of files you don't know what to do with.

THE BUILD LOOP

Building a tool this way is a loop, not a one-shot. You describe it, the AI generates it, you actually use it, you tell it what's wrong, it fixes it, and you ship. The "use it and tell it what's wrong" step is where the quality comes from. You don't need to know how it works — you just need to notice what's off and say so, like reviewing a contractor's work before you sign off.



The loop, not a one-shot. Most of the value is in step 4 — trying it, then telling it plainly what to change.

The trick is starting with a clear first prompt. Be specific about **what it does, what goes in, and what comes out.** Here's a starter you can copy and adapt:

```
Build me a single-file HTML quote calculator.
Inputs: service type (dropdown), estimated hours, hourly rate,
and a "rush job" checkbox that adds 20%.
Show the total updating live as I type, in big clear text.
Add a "Copy quote" button. Keep the design clean and simple.
```

Then you use it, and you refine in plain English — no code talk needed. You just describe what's off, one change at a time:

```
// refine by describing, not coding
Make the total bigger and green.
Add a field for a discount percentage.
Put "+20% rush fee" next to the rush checkbox.
Round the total to the nearest dollar.
```

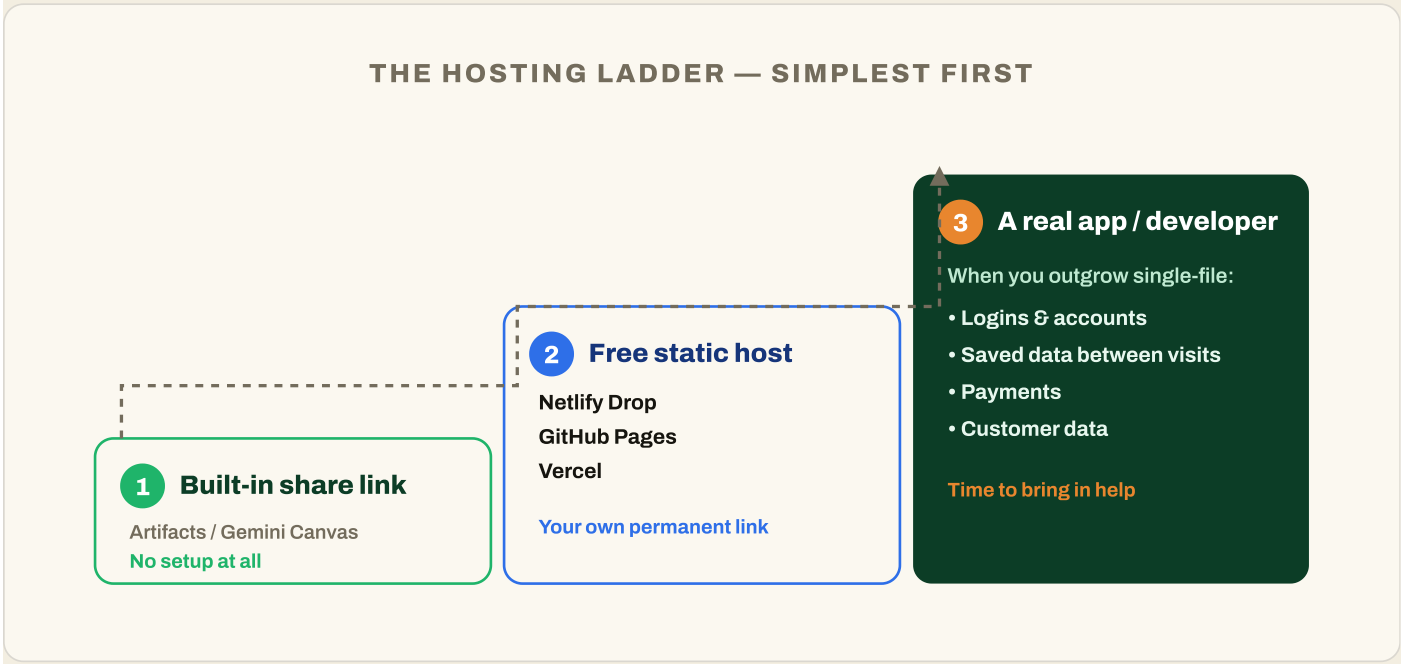
Keep going until it does exactly what you want. Each message just adjusts the same tool — nothing starts over. When it's right, you **ship**, which for a personal tool might mean bookmarking it, and for a shared one means grabbing a link (that's next). The whole loop for a small tool is usually minutes, not hours.

ONE CHANGE AT A TIME

If you pile ten fixes into one message and something comes back wrong, you won't know which change caused it. Adjust a couple of things, check that it still works, then keep going. It feels slower and is actually faster.

WHEN YOU WANT A REAL LINK: THE HOSTING LADDER

For a tool only you use, you don't need hosting at all — the file on your computer, or the share link from Artifacts or Canvas, is plenty. But when you want a real, always-on link to send around, here's the ladder, simplest and cheapest first. Climb only as high as you actually need.



Start at the bottom. Most business tools never need to leave rung one or two. The jump to rung three is a real project — know when you're taking it.

1. **The built-in share link.** Artifacts and Gemini Canvas both hand you a link the moment your tool works. Zero setup. Perfect for "send this to a couple of coworkers" — which covers most internal tools.
2. **Free static hosting.** When you want your own permanent link, **Netlify Drop** lets you literally drag your HTML file onto a web page and get a live URL back — no account needed to start. **GitHub Pages** and **Vercel** do the same with a little more setup, and all of them are free for something this small.
3. **A real app, or a developer.** When the tool needs to save data between visits, have user logins, handle payments, or hold customer information, you've outgrown single-file HTML. That's your signal to bring in a developer or move to a proper app platform. Knowing where that line is saves you from trying to force a real product out of a quick tool.

OPTION	SETUP EFFORT	REACH FOR IT WHEN
Share link (Artifacts / Canvas)	None	Sharing with a few people, right now
Netlify Drop	Drag the file in	You want your own link, no account fuss
GitHub Pages	A little (free account + repo)	You already use GitHub or are a bit technical
Vercel	A little (free account)	You want a fast, polished URL you might grow
Developer / real app	A real project	Logins, saved data, payments, customer data

DON'T OVER-BUILD

The most common mistake is jumping to rung three too early — trying to turn a handy internal calculator into "a real app" before anyone's even using the simple version. Ship the single file, let people use it, and only climb the ladder when a real limit actually bites.

FIVE PROMPTS YOU CAN STEAL TODAY

Enough theory. Each of these builds a genuinely useful internal tool. Copy one, paste it into Claude or Gemini, and refine from there. Notice the pattern in all of them: name the tool, list the inputs, say what comes out, and ask for a single self-contained file.

1 · Cost-of-a-meeting calculator

The one that changes behavior. Put a real dollar figure on that recurring meeting and half your calendar cleans itself up.

```
Build a single-file HTML "cost of a meeting" calculator. Inputs: number of attendees,
average hourly cost per person, meeting length in minutes, and how many times a month it
repeats. Show the cost of one meeting and the yearly total in big, clear numbers.
```

2 · ROI calculator for a purchase

Hand this to anyone pitching you a new tool or subscription. If the numbers don't clear the bar, the conversation's over fast.

```
Build a single-file HTML ROI calculator. Inputs: monthly cost, hours saved per week, and the
hourly value of that time. Output the monthly savings, the payback in weeks, and a plain-
English verdict like "worth it" or "not yet."
```

3 · Proposal generator

Stop rewriting the same proposal from scratch. Fill in the boxes, copy the result, tweak the wording, send.

```
Build a single-file HTML proposal generator. Inputs: client name, project scope, timeline,
and price. Output a clean, formatted proposal with a header, a scope section, a timeline,
and the total, plus a "copy to clipboard" button.
```

4 · Onboarding checklist app

Every new hire, every new client — the same steps, done the same way, nothing forgotten. This is the tool that makes you look organized.

```
Build a single-file HTML onboarding checklist. Show a list of steps with checkboxes, a
progress bar that fills as items are checked, and a "reset" button so I can reuse it for the
next person.
```

5 · Simple KPI dashboard

For the numbers you check every week. Paste them in, see them laid out cleanly, and stop hunting across five tabs.

```
Build a single-file HTML dashboard. Let me type in a handful of key numbers — revenue, new
leads, churn, cash on hand — and lay them out as clean cards with big figures and labels.
Keep it simple and easy to read at a glance.
```

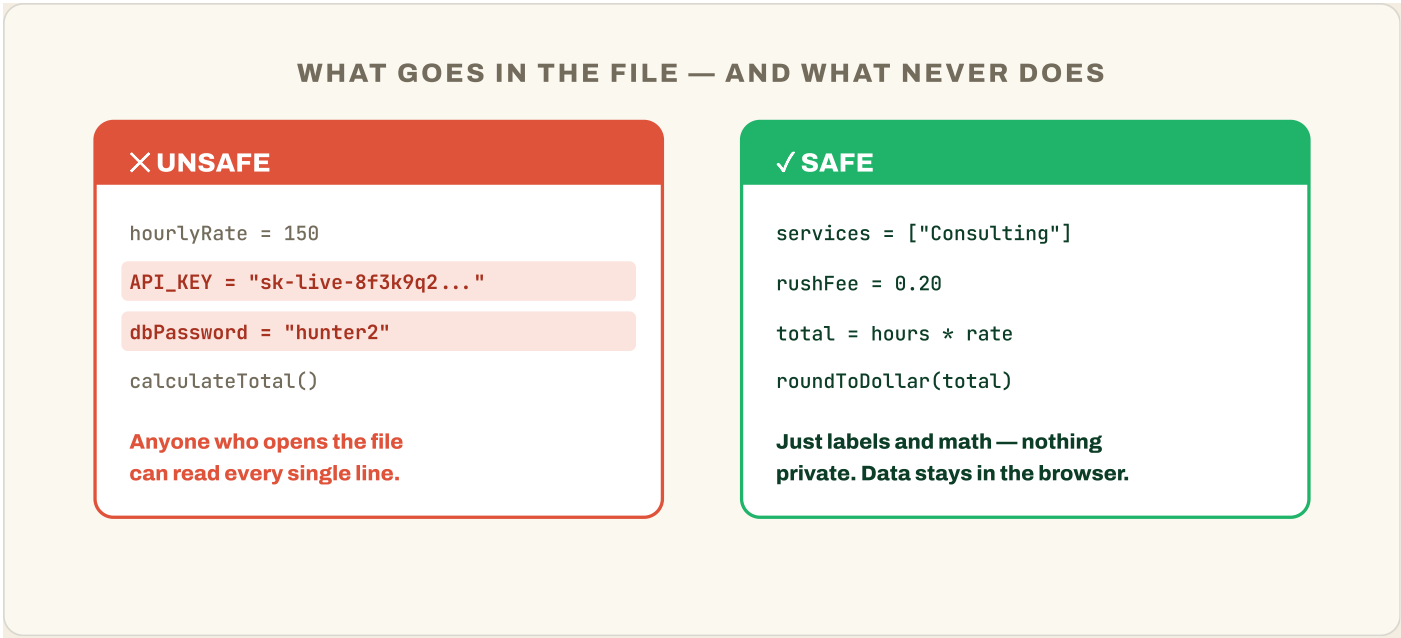
TWO GUARDRAILS THAT KEEP THIS SAFE

Get these two things right and you'll never have a problem. Both are simple once someone points them out — which almost nobody does.

1 · Never put secrets in the file

Remember that a single HTML file is **just a readable text file** — that's the whole reason it works without a developer. But it also means anything you type into it can be read by anyone who opens it. So never put a password, an **API key**, a bank detail, or any private credential inside a client-side HTML tool. If a tool genuinely needs a secret to work, that's the sign it needs a proper backend — a developer's job, not a single file.

Quick translation, since the term comes up: an **API key** is basically a password that lets one piece of software use another — like a key to your email account or your payment system. Sitting in a plain HTML file, it's in plain sight to anyone who opens the page. Keep them out.



Left: secrets in a file anyone can read — never do this. Right: labels and math only, with data staying in the browser. That's the safe shape of a no-code tool.

2 · Know internal vs customer-facing

There's a big difference between a tool for you and your team, and something you put in front of customers. A personal or internal tool can be a little rough — it only has to work for people you trust. The moment strangers use it, or it touches customer data, the bar jumps: privacy, reliability, and security all start to matter, and that's developer territory. Be honest with yourself about which one you're building, and don't quietly let an internal tool drift into a customer-facing one.

WHERE YOUR DATA LIVES

By default, a single-file tool keeps everything **in your browser** — nothing is sent anywhere, which is great for privacy. But it also means the tool doesn't remember anything after you close the tab, and there's no shared database. If you need data saved or shared across people, you need a backend — and that's exactly where "no code" ends and a developer begins.

WHAT TO DO NEXT

Pick the smallest tool on your wish list — the quote calculator is a perfect first one. Open Claude, paste the starter prompt from the build-loop section, and refine it until it's yours. You'll have a working tool before your coffee's cold, and you'll never look at "we should build a quick tool for that" the same way again.

SOURCES & WHERE TO LEARN MORE

- ↗ Anthropic — What are Artifacts and how do I use them?
- ↗ Anthropic — Build and share AI-powered apps with Artifacts
- ↗ Google — Gemini Canvas overview
- ↗ MDN — What is HTML? (plain-English intro)
- ↗ Netlify Drop — drag a file, get a live link
- ↗ GitHub Pages — free static hosting
- ↗ Vercel — free hosting for small sites

Built by Hank Barker // AI Essentials // @hankhelpsai. These tools ship changes constantly — last reviewed July 2026.