CLAUDE CODE · 16 min read · August 4, 2026

The Claude Code Cheat Sheet That Puts You Ahead of Everyone Else

Claude Code isn't just a coding tool. Here's exactly how each of its 10 most useful features actually works, with real before/after examples for people who've never written a line of code.

Most people who've heard of Claude Code think of it as a coding tool. It's not just that. It's a tool that can take real action on your actual computer, files, folders, research, and business documents, not just code. Most beginners open it, use it like a slightly smarter chatbot, and never touch the other nine things on this list. Here's exactly what each one does, and how to actually use it.

GROUP 1 What Makes It Different

1. Tool Use

A regular AI chat ends with words. You get an answer, and you're the one who has to go do something with it. Claude Code doesn't stop at the answer, it can run the next step itself, then check whether that step actually worked, and adjust if it didn't.

EXAMPLE OUTPUT

You ask a regular chatbot: "How do I rename 40 client photos to match our naming convention?" You get back a set of instructions, and you still have to go do it yourself, one file at a time.

You ask Claude Code: "Rename every photo in this folder to ClientName-Date-Number." It opens the folder, reads what's actually in there, renames all 40 files to match, and then tells you what it did, "Renamed 40 files, here are the first 5 so you can check the pattern is right." The work is already done, not just explained.

TIP

This is the single biggest mental shift for beginners. You're not writing a message to get an answer, you're giving a direction and letting it carry it out.

2. File Access

Claude Code can read, edit, organize, and create real files directly on your computer. It’s not limited to code files either, it can open Word documents, PDFs, spreadsheets, and plain text files, actually read what’s inside them, and act on that.

Real things this makes possible for non-coding work:

- Go through a messy Downloads folder and sort everything into real subfolders by type or date
- Open 15 separate client PDFs and pull the same 3 pieces of information out of each one into a single summary spreadsheet
- Read a rough, disorganized set of notes and turn it into a clean, formatted document, saved exactly where you tell it to
- Find every file that mentions a specific client or project across a whole folder, without you opening each one

TIP

File Access plus Tool Use together are what make Claude Code fundamentally different from ChatGPT's regular chat window: those two features are what let it do the work instead of just describing the work.

GROUP 2

How to Actually Talk to It

3. Better Prompts

This one isn't a Claude Code feature exactly, it's a practice, but it matters more here than anywhere else, because Claude Code takes real action on real files. A vague prompt to a chatbot gets you a vague answer you can just ignore. A vague prompt to Claude Code can get you a real, unwanted change to a real file.

EXAMPLE OUTPUT

Vague: "Clean up my website copy."
Claude Code has to guess what "clean up" means, how much to change, and what you'd consider too far.

Specific: "On the homepage only, fix typos and grammar. Don't rewrite any sentences or change the tone. Show me a list of every change before you save anything."
Same request, but now there's no guessing: what to touch, what to leave alone, and what the finished result should look like are all spelled out.

Three things worth including every time: what you want, what you explicitly don't want touched, and what "done" looks like. Brief it the way you'd brief a new employee on their first day, not the way you'd type a question into a search bar.

Steal this template and fill in the blanks any time you're about to ask for something that touches real files:

PROMPT

```
I want you to: [the actual task]

Do not: [anything that's off-limits, files/sections/tone to leave alone]

When you're done, show me: [a list of changes, a summary, the finished file, whatever "done" means here]
```

4. CLAUDE.md

This is a real file you create once, named exactly `CLAUDE.md`, saved in the main folder of whatever project you're working in. Claude Code checks for this file automatically at the start of every session in that folder, before you've typed a single word, so your rules, preferences, and context are already loaded.

EXAMPLE OUTPUT

A real CLAUDE.md for a small business's content folder might look like:

"This folder is for [Business Name]'s social media captions. Voice: warm, never salesy. Never use the word 'delicious.' Always pull current prices from `menu.md` in this folder, never guess a price. If a detail about an ingredient or allergen is missing, ask before writing anything. Every caption ends with a question to encourage comments."

Every single time Claude Code works in that folder, it already knows all of that, automatically, without being told again.

Don't stare at a blank file. Paste this prompt in and answer its questions, and Claude Code will write your first CLAUDE.md for you:

PROMPT

```
Help me write a CLAUDE.md file for this folder. Ask me: what this folder/project is for, any
rules or
preferences I want followed every time (tone, formatting, things to never do), and any facts
you should
already know (business details, standing context) instead of me repeating them. Then write the
finished
CLAUDE.md file, clear and short, no filler.
```

TIP

You can also have a personal, account-wide version of this that applies everywhere, not just one project folder. Start with a project-level one first, it's the easier place to see the effect immediately.

5. Plan Mode

Plan Mode makes Claude Code stop and show you exactly what it's about to do, step by step, before it changes anything. You read the plan, adjust it, or approve it, and only then does the actual work start.

EXAMPLE OUTPUT

You ask it to reorganize a shared project folder. In Plan Mode, before touching a single file, it shows you something like: *"1. Create three new folders: Contracts, Invoices, Reference. 2. Move 12 PDF files into Contracts based on filename matches. 3. Move 8 files into Invoices. 4. Leave 4 files unsorted, flagged for you to review, their names don't clearly match a category."* You can say "skip step 4" or "actually call it Billing not Invoices" before anything actually moves.

This exists for exactly the moment where you want the power of Tool Use and File Access, but you're not ready to just hand over the keys yet, especially useful the first few times you try a new kind of task.

GROUP 3

Leveling Up

6. Skills

A Skill is a saved, reusable set of instructions for a specific task you do more than once, built one time and then triggered with a short command instead of re-explaining the whole process from scratch every time.

EXAMPLE OUTPUT

Say you format every blog post the same way: a specific title style, an intro paragraph under 3 sentences, subheadings every 200 words, and a closing call-to-action. Instead of typing all of those rules out every time, you save them once as a Skill. Next time, you just say "format this post" and Claude Code applies the whole saved process automatically.

TIP

Don't confuse this with Claude Cowork's "Record a Skill" feature (the screen-recording one covered in an earlier guide), that's a different product entirely. Claude Code's Skills are written instructions you build and reuse inside your coding/file workflow specifically, no screen recording involved.

7. MCP

MCP (Model Context Protocol) is what lets Claude Code connect directly to outside tools and business apps, Google Drive, Slack, Notion, Airtable, Gmail, and a long list of others, instead of staying limited to just the files on your computer.

EXAMPLE OUTPUT

Without MCP: you'd have to open Google Drive yourself, find the right document, copy the text out, and paste it into your conversation with Claude.

With Google Drive connected through MCP: you just say "pull the client brief from my Drive and summarize the key deadlines," and it goes and gets the real document itself, no copy-pasting, no switching windows.

Each connection is something you set up once (an authorization step, similar to logging into an app), and after that it’s just available in conversation whenever you need it.

8. Memory

Claude Code doesn’t have some hidden, automatic memory quietly learning everything about you in the background. It remembers because you tell it to, usually by having it write something down, a correction, a preference, a standing rule, into your CLAUDE.md file or a dedicated notes file, which then gets read back in automatically every future session.

EXAMPLE OUTPUT

You correct it once: "Don't use em dashes in anything you write for me." If that correction only lives in the current conversation, it's gone once you start a new one. But if you say "remember that, write it to CLAUDE.md," it's saved permanently, and every future session already knows the rule before you've said anything.

TIP

The honest version: memory isn't magic, it's just a file that gets read automatically. It's only as good as what you actually tell it to save, so when you correct something you want remembered, say so explicitly.

GROUP 4

Working Smart, Not Slow

9. Context

Every conversation has a limited working memory, called the context window, everything you've discussed so far has to fit inside it. On a long session, that window fills up, and once it's close to full, responses can get slower or lose track of earlier details.

Two real commands manage this:

- `/compact`: compresses the conversation down to what still matters, keeping the project going without losing the thread. Use this mid-project, when you're deep into something and want to keep working.
- `/clear`: wipes the conversation completely for a genuinely fresh start. Use this when you're switching to something unrelated, not partway through the same task.

> TIP
>
> A simple rule of thumb: if you're still working on the same thing, `/compact`. If you're moving to something new, `/clear`. Starting a new, unrelated task without clearing is the most common way beginners end up with confused, unfocused responses.

10. Permissions

You control exactly how much freedom Claude Code has before it acts, and it's not one all-or-nothing switch.

- **Ask before everything**: it proposes each file edit or command and waits for your yes before doing it. Safest starting point for a first project.

- **Auto-accept edits**: file changes happen without asking each time, but it still checks with you before anything riskier, like running a command that could affect something outside the project.
- **Full autonomy**: it handles routine work start to finish without stopping to ask. Best saved for tasks and folders you already trust it with.

You can set different permission levels for different kinds of actions, it's genuinely adjustable, not just one setting for everything. Most people start cautious and loosen it gradually as they see what it actually does with the freedom they've already given it.

The takeaway

Most beginners use Claude Code like a slightly smarter chat window and never touch the other nine things on this list. The real value shows up once you start treating it like what it actually is: something that takes real action on your actual work, remembers what you tell it to, connects to the other tools you already use, and gives you real, adjustable control over how much it's allowed to do without asking first.