

Claude Code

The cheat sheet - from your build session, right next to your keyboard

OCTOBER
2026

KEEP THIS OPEN IN A TAB OR PRINT IT · SAVES YOU SEARCHING THOSE FIRST WEEKS

1 GETTING STARTED

- 1 Open your **terminal** in your project folder.
- 2 Type **claude** and hit enter.
- 3 Run **/init** once - Claude reads your project and creates an instruction file. Name it **AGENTS.md** and other AI tools read it too.
- 4 Run **/statusline** once - from then on you always see how full your context is.
- 5 Then just ask, in plain English. Build, explain, fix - anything goes.

AGENTS.md = your project's memory. Put agreements, style and ground rules in here; Claude reads it automatically every session.

3 KEYS TO KNOW BY HEART

- Shift** + **Tab** Switch permission mode (see block 5)
- Esc** Interrupt Claude mid-answer
- Esc** **Esc** Open the rewind menu
- @** Reference a file or folder
- ↑** **↓** Browse your previous prompts
- Ctrl** + **R** Search back through your prompt history
- Ctrl** + **V** Paste an image or screenshot (Mac: Ctrl, not Cmd. Windows: Alt+V)
- Ctrl** + **C** **Ctrl** + **C** Twice = quit Claude Code
- ?** Show all shortcuts

4 MCP & CLI - YOUR NEW COLLEAGUES

- MCP** *An expert colleague you install* - knows the documentation, operates a service's API. E.g. the Cloudflare MCP (block 8).
- CLI** *Command line interface* - tools driven by text commands; Claude talks to these at lightning speed. E.g. the GitHub CLI (gh).

2 THE COMMANDS YOU'LL ACTUALLY USE

- /clear** Start fresh. *New topic = clear first.*
- /compact** Shorten a long conversation without losing the thread.
- /usage** What you have used and how much of your plan is left.
- /rewind** Go back to an earlier point. *Wrong turn? Rewind.*
- /resume** Pick up an earlier conversation.

5 THE 4 MODES (SHIFT+TAB)

- **Manual** - politely asks permission for every action. Safe starting point.
- **Accept Edits** - approves changes automatically. Use once you trust it.
- **Plan** - read-only. Thinks along, touches nothing. Ideal before a big step.
- **Auto** - keeps going without asking each time. The starting mode in recent versions.

6 HOW TO GET THE MOST OUT OF IT

- 1 Give **context**, not just an instruction: what, what for, for whom.
- 2 For big things, switch to **plan mode** first (**Shift** + **Tab**) - read the plan, then approve.
- 3 Work in **small steps**. Finished a piece? **/clear** and move on.
- 4 Unsure about a change? **Esc Esc** and rewind. Nothing breaks.
- 5 Slop alarm: ask for **character**, give examples. "Don't make it generic."

7 SKILL OR AGENT?

	SKILL	AGENT
WHAT	Knowledge + instructions in a file (SKILL.md): what AI <i>knows</i> .	An acting entity that executes, decides, iterates: what AI <i>does</i> .
NATURE	Passive - loaded when relevant.	Active - a chain of steps: input → logic → output, with tool access.
E.G.	Your house style, review checklist, quoting conventions.	A helper that works out a subtask while you carry on.

Rule of thumb: knowledge you want to **reuse** → skill. A job that must run **on its own** → agent. They combine well: an agent loads skills.

8 THE STACK TO BUILD WITH

Nuxt *Framework* - site, app and API in one project. Claude knows it inside out.

Tailwind *Styling* - design as utility classes. AI writes it fluently and consistently.

Cloudflare *Hosting* - Pages & Workers, free to start. Hook up the Cloudflare MCP and you're live in seconds.

GitHub *Version control* - every step recorded; Claude drives it via the GitHub CLI (gh). Home base for your skills (block 9).

Tip: create a **design.md** with your brand style and assets, and share it like your skills (block 9) - everything you build is instantly on-brand.

9 SHARING SKILLS ACROSS YOUR ORGANISATION

① Put your skills in **one git repo**:
skills/<name>/SKILL.md.

② Everyone clones it, e.g. to ~/git/skills.

③ Link with a **symlink** from the git folder: `ln -s ~/git/skills/<name> ~/.claude/skills/` - the skill works in every project and the repo stays the source.

④ Colleague built something new? `git pull` and you have it too.

10 SUPERPOWERS

/skills *Skills* = ready-made recipes. One command, whole job done (e.g. a GEO/SEO check on your site).

/skill-creator *Claude helps you make a skill*. Install once: `/plugin install skill-creator@claude-plugins-official`

/agents *Subagents* - ask Claude for a helper that works out a subtask while you carry on.

/mcp Connect external sources (Stripe, GitHub, your analytics) so Claude can reach them directly.

/memory Edit what Claude remembers about your project.

/model Switch models (e.g. a heavier model for hard jobs).

/voice *Talk instead of type* - sharing thoughts out loud beats typing them. Great for brainstorming and giving context.

11 WHICH MODEL WHEN?

Opus 5.5 *The default* - strong on everyday and complex work. Start here.

Sonnet 5.5 *The workhorse* - routine tasks, fast and still very capable. Switch: `/model`.

12 THINKING LEVEL (EFFORT)

/effort Dial how thoroughly Claude thinks: *low* → *medium* → *high* → *xhigh* → *max*. Simple job? Lower saves time and money. Default: *medium*.

ultrathink Type it right in your prompt: *one-off extra-deep thinking* for that answer, without changing the session.

13 BUILD A SECOND BRAIN

① **Transcribe everything** - meetings, calls, loose thoughts. Dictation: Paraspeech (Mac), Oorapp (free, beta) or Handy (Windows, free). As a team: Wispr Flow or Fireflies (paid, cloud).

② Store the transcripts as text files in a folder in your project.

③ **Feed them to your agents** - every meeting becomes context to build with, every thought searchable.

Recording → transcript → context → build. Your conversations become **fuel** instead of memories.