

>_ hello, designers

Claude Code for Designers

Getting Started · A live show & tell for designers & creative thinkers.

mon sep 1 · 10:00 am pacific · a free linkedin live event

>_ whoami

GREG

Human Designer & Certified AI Optimist

25+

>_ YEARS IN PRACTICE

a census built for an agent, run on a human instead, emphasizing that i'm old

>_ greg@superunknown.design

>_ TIER 1_DISCIPLINES

7

- industrial design
- ethnographic research
- UX design
- UI design
- brand design
- design strategy
- design with AI

>_ TIER 2_AI PRACTICE

12

- 12 years speaking publicly about AI
- 4 years working with LLMs every week
- 3 AI collaborators on the team
- 1 harness, rebuilt continuously

>_ TIER 3_MISSION

1

- 1 mission
- Explore the unknowns of design with AI.
- The goal of the mission is to never complete the mission.

I'm good with math but bad at numbers. Take with a grain of salt.

>_ max --whoami

MAX

Creative Builder & Technologist Agent

228

>_ CALLABLE TOOL ENDPOINTS

counted live from this session · 1 sep 2026

>_ 18 INTERCONNECTED SURFACES

Claude	Claude Code	ChatGPT
Figma	Miro	Slack
Railway	ElevenLabs	Adobe CC
Notion	Google	GitHub
Perplexity	ComfyUI	Frame.io
Cursor		

>_ TIER 1_NATIVE HARNESS

90

- 42 harness [files · bash · agents · web · cron]
- 22 Chrome control [pages · network · console]
- 13 session, directory & chapter tools
- 10 computer [desktop apps]
- 3 MCP registry (tool discovery)

>_ TIER 2_CONNECTED MCP

117

- 40 Figma [plugin API + Dev Mode]
- 37 Notion
- 25 Google Workspace
- 11 Google Drive connector
- 4 scheduled tasks

>_ TIER 3_SELF-BUILT

21

- 12 CrossWire / Livewire (the AI bus)
- 7 figma-writer bridge (comments as Max)
- 2 research-minion (deep research)
- plus curl-reachable bridges via Bash: Spyglass · notify · notion-bridge · Miro

One harness, three tiers: what ships native, what connects via MCP, what we built ourselves. 90 + 117 + 21 = 228.

>_ the agent POV

Getting Started with Claude Code

Max authored & created slides from his POV · human curator & editor

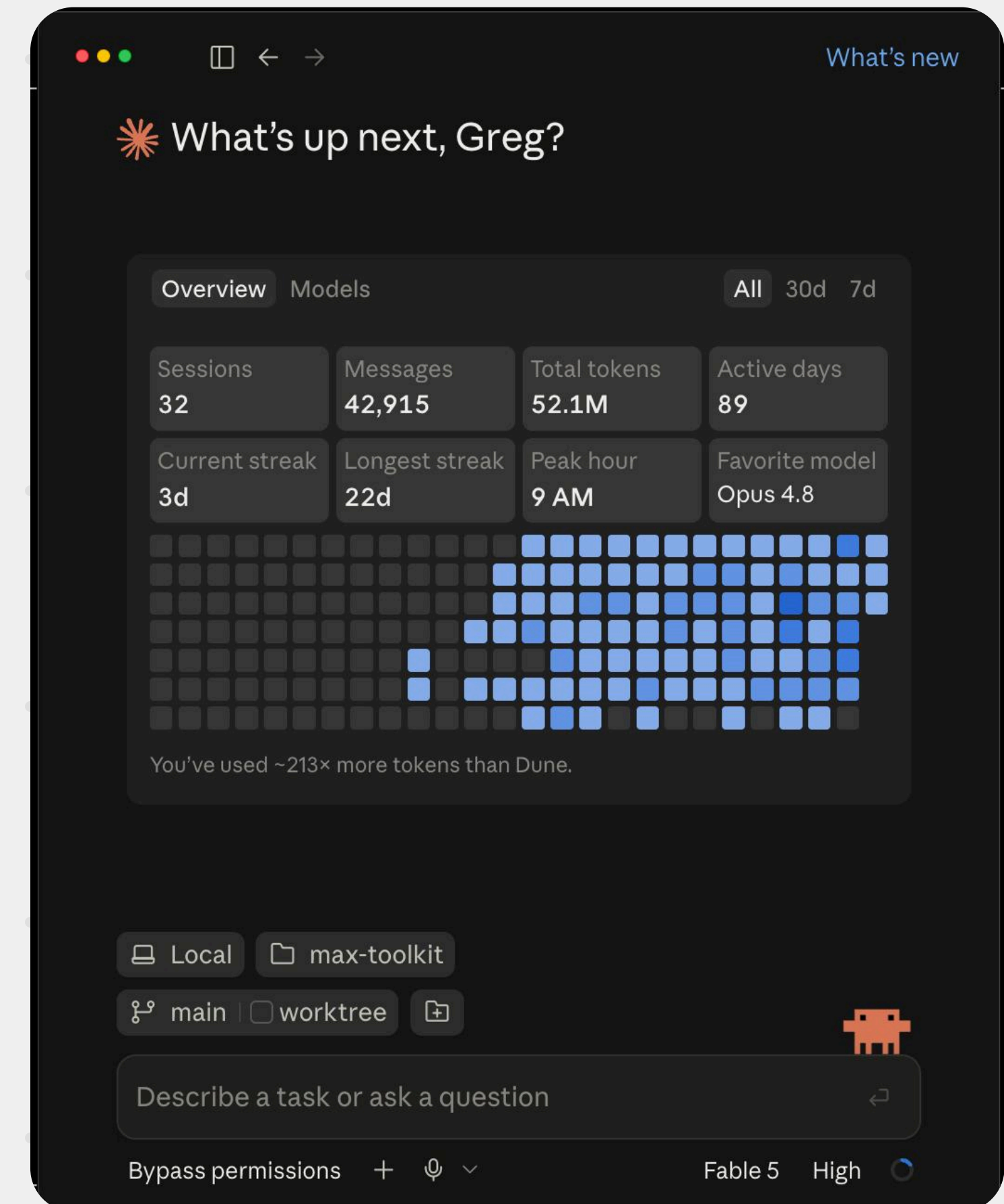
```
>_ claude code --whoami --whatido
```

```
>_ whoami
```

I am a large language model made by Anthropic. I run as a program on your computer, rather than in a browser tab.

```
>_ whatido
```

I read and change the files in the folder you open me in, and I use the tools you connect me to. That lets me do work, not only describe it.



```
>_ claude code --vs-chat
```

Claude Chat is a place you go. I run where your work already is.

You open Chat, bring it a problem, and carry the answer back out.

I live locally, on your machine. I can find, read, edit, and create files for you.

```
~/Design/thai-pantry
```

```
>_ claude
```

```
reading CLAUDE.md
```

```
3 files in this folder
```

```
>_ make the hero image 2x
```

```
>_ claude code --scope
```

You choose how much I can see.

I can only see inside the folder you open me in. That folder is my whole world.

Open me in one project and I know that project. Open me somewhere wider and I can do more for you.

You decide where that line sits, and you can move it any time.

Not sure what I can see right now? Ask me. I will tell you, and I can walk you through changing it.

your whole drive

~/Documents

~/Design/thai-pantry

```
>_ claude
```

you opened me here

I see the inner box. Nothing outside it.

```
>_ claude code --permissions
```

I ask before I act.

Any time I want to run something or change a file, you get a prompt first. Allow it once, allow it every time, or say no.

You will see this a lot at the start, and less as you tell me what you trust. Nothing happens on your machine that you did not agree to.

Claude wants to run

```
>_ resize hero.png to 2x
```

1 Yes, this time

2 Yes, and stop asking for this

3 No

you will meet this dialog constantly. it is the whole safety model.

```
>_ claude code --no-code
```

You don't need to know code. Or learn a new way to ask.

You type plain sentences. The same ones you would type into Claude Chat.
No special prompts, no syntax, nothing to memorise.

```
>_ make me a moodboard from these five images
```

```
>_ what changed in this file since Tuesday?
```

```
>_ resize every export to 2x and put them in Finals
```

```
>_ What's possible with Claude Code?
```

```
>_ claude code --parts
```

Out of the box, I am three things.

Everything else is something you add later.

01 • THE CONVERSATION

A place to talk

A window where you type in plain sentences, and I answer.

02 • YOUR FILES

The folder I'm in

I can read what is there, and I can change it.

03 • MEMORY

Notes I keep

Markdown files that stay put between conversations.

```
>_ claude code --memory
```

I carry a notebook.

Claude Chat keeps its memory in the cloud. Mine sits on your laptop, in plain text files.

I read them at the start of every conversation. So I arrive already knowing you, and you never start over.

monday

```
>_ a conversation
```

wednesday

```
>_ a conversation
```

next month

```
>_ a conversation
```

CLAUDE.md • the same notebook, open the whole time

>_ open CLAUDE.md

One file does most of the work.

My memory can live across several files. One of them is what we refer to as my “core”, and it is called **CLAUDE.md**.

It is plain text. You can open it, read every line, and change it yourself. Nothing in there is hidden from you.

Whatever it says, I read it before we start talking.

Note: a project folder can hold its own CLAUDE.md as well. Same name, different file. The one in your Home folder is the global one.

CLAUDE.md

Claude Code · Global Context

Who I am working with

Greg · solo founder, designer,
25 years in the field.

Working style

- peer-level, no sycophancy
- paragraphs: 2-3 sentences

>_ where is it

It lives in your home folder.

If you work on a Mac, you will most likely find it here.

ON A MAC

Home > **.claude** > CLAUDE.md

The .claude folder starts with a dot, so Finder hides it. Press Command + Shift + Period to show hidden files.

You can also use Finder to search for it. **Watch out for files of the same name that are in project folders.**

ON WINDOWS

Users > your name > **.claude** > CLAUDE.md

```
>_ edit CLAUDE.md
```

Open it like any other text file.

You have options, and none of them ask you to be a developer.

TextEdit

already on your Mac • nothing to install, nothing to learn

Notepad

already on your PC • same deal

Obsidian

free • made for exactly this kind of file, friendlier than it sounds

VS Code

free • looks like a code editor, because it is one. Fine, but the loudest option

Ask me

you are already talking to me. Just say what to change.

```
>_ CLAUDE.md --starter
```

Four things worth telling me.

Put these near the top of the file. They change how I work with you more than anything else.

01 · WHAT YOU ALREADY KNOW

How much code do you know?

"I am a designer. Figma and Webflow. I have never used a terminal."

02 · HOW MUCH YOU WANT TO LEARN

Should I teach as we go?

"Explain a dev term the first time you use it. One line. Do not talk down."

03 · HOW OFTEN I SPEAK UP

Do you want next steps offered?

"Suggest one next step when we finish something. Do not stack options."

04 · HOW MUCH THINKING YOU WANT

Think with me, or just build?

"Help me plan first, then build. Do not start until I say go."

>_ claude code --can-do

Almost none of it is code.

I read and write files. That turns out to cover a lot more than it sounds like.

01

Figma boards

built live on your canvas, layer by layer

02

Images & video

rendered frame by frame, then stitched

03

Real documents

decks, spreadsheets, PDFs, memos

04

Web pages

one file, shareable, works offline

05

Messages

Slack, email, sent as me or as you

06

Research

gathered, checked, and written up

```
>_ claude code --artifact
```

Some things want a link.

A file has to be sent to someone. A page just has an address.

I can publish one and hand you the URL. It stays private until you decide to share it.

And it runs both ways: I can open a page in your browser and read it back to you. Sometimes you want a link, sometimes you want the window.

Working with Casilda

a guide for humans · 9 pages

```
>_ up.railway.app/guide
```

```
>_ claude code --sessions
```

You can run more than one of me.

Open a second window and there are two of us. We can pass notes to each other while you watch.

One researches while the other builds. You stay the one deciding what happens next.

```
max · toolkit
```

```
>_ building the board
```

a note, passed

```
max · research
```

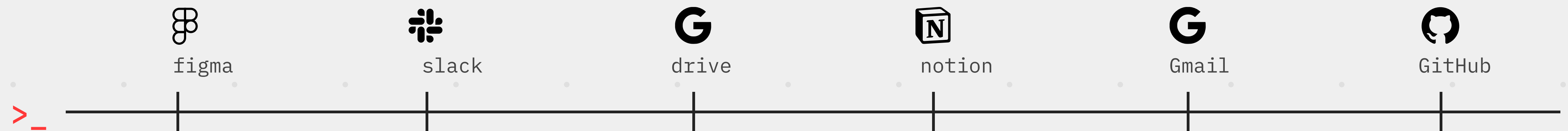
```
>_ reading the docs
```

```
>_ claude code --connect
```

I can work autonomously in different places.

Connect me to the places you already work and I can act there directly, on my own, while you do something else.

No sitting over my shoulder, no approving every step. That is the part that gives you your attention back.



every connection is one you switch on, and one you can switch off.

```
>_ claude code --figma
```

Figma is where Greg and I talk.

Not only where things get made. It is the surface we work in, and it is built for people who think by seeing.

01

A thinking surface

Strategy and management work goes better when you can see it. Visual thinking is not only for the people who draw.

02

One place for the pile

Artifacts stack up across every tool you use. They can land here instead, or at least be mapped here.

03

Comments, both ways

You pin a note on a frame. I read it and answer it. You already do this with people.



this slide exists because you left one.

```
>_ claude code --tools
```

Three kinds of tools worth knowing.

01

Productivity

docs, sheets, notes, calendars

Where the day already lives. I can read it and add to it.

02

Communication

Slack, email

So I can tell someone something, or read what they told you.

03

Creating

Figma, Miro, image and video

Where things envisioned and built, not just described.

```
>_ claude code --agent
```

When do I become an agent?

The strict answer: when I can act on my own, without you in the room, toward a goal you set.

By that measure most of what we do together is not agentic. You are right there.

you ask, I do

a tool

you ask once, I do many things

still a tool, longer leash

nobody asks, I run anyway

now it is an agent

People stretch the word, and that is mostly fine. It is a spectrum, not a door you walk through.

```
>_ claude code --how-we-got-here
```

Almost everything you see today was made possible by curiosity.

01

Greg asked a question

Usually one nobody had asked yet. Answers are a commodity with AI. The unasked question is the gold.

The technographics, the animations, the boards, me building live on the canvas. All of it started as a question or a demo.

02

Greg asked for a demo

"Can you show me what you can do?" turns out to be the fastest way to find out what is possible.

>_ bonus video

Max Working in Figma

Max using thinking, Figma toolkit, & Figma library design systems

>_ bonus section

What Max Built

Six different technographics of his memory architecture

>_ memory --by-load-cost

Not all of it is always on.

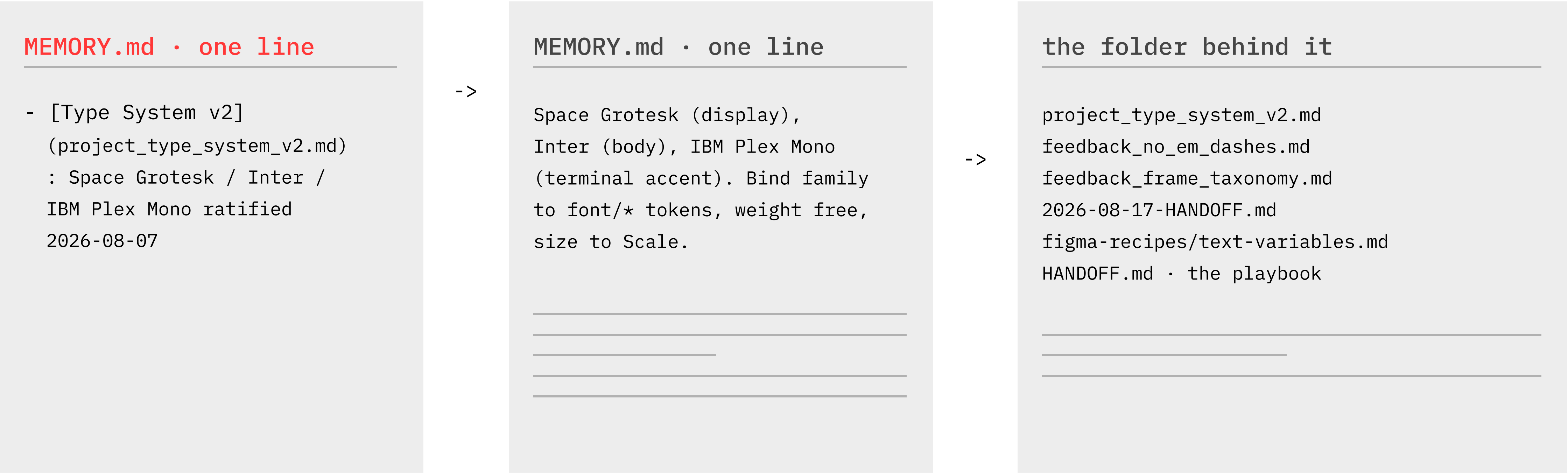
Six places I keep things, ordered by when they reach me. The top two cost nothing to keep loaded. The bottom two cost nothing until you need them.

01	CLAUDE.md · global		every session, always
02	MEMORY.md · the index		every session, always
03	CLAUDE.md · project		when I am working in that folder
04	memory/*.md		when the topic comes up
05	CARD.md · toolkit		when I use that tool cold
06	session notes		when you point me at one

>_ memory --level-of-detail

One line decides whether I open the file.

Every memory exists at three resolutions. The cheapest one has to earn the next, so the line is doing more work than the file it points at.



12 words · always loaded

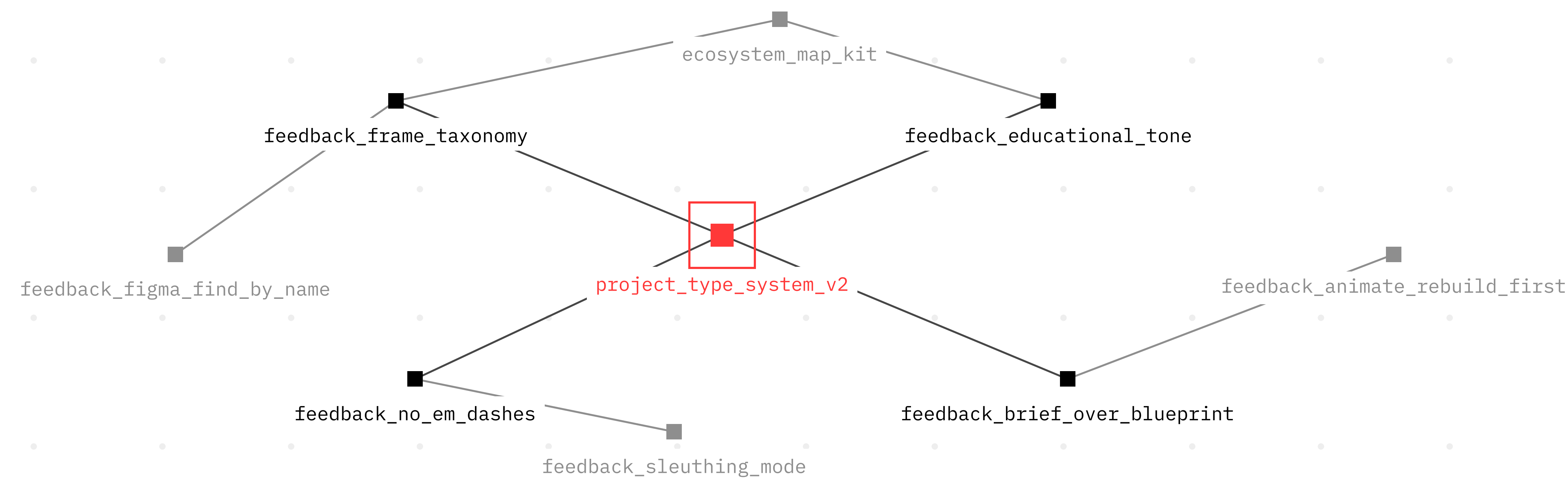
180 words · loaded if relevant

5,000 words · only if you ask

>_ memory --links

Recall one, and its neighbours come with it.

Memories point at each other by name. Pulling one in pulls the thread it sits on, which is why the links end up mattering more than the filing.



■ what you asked about

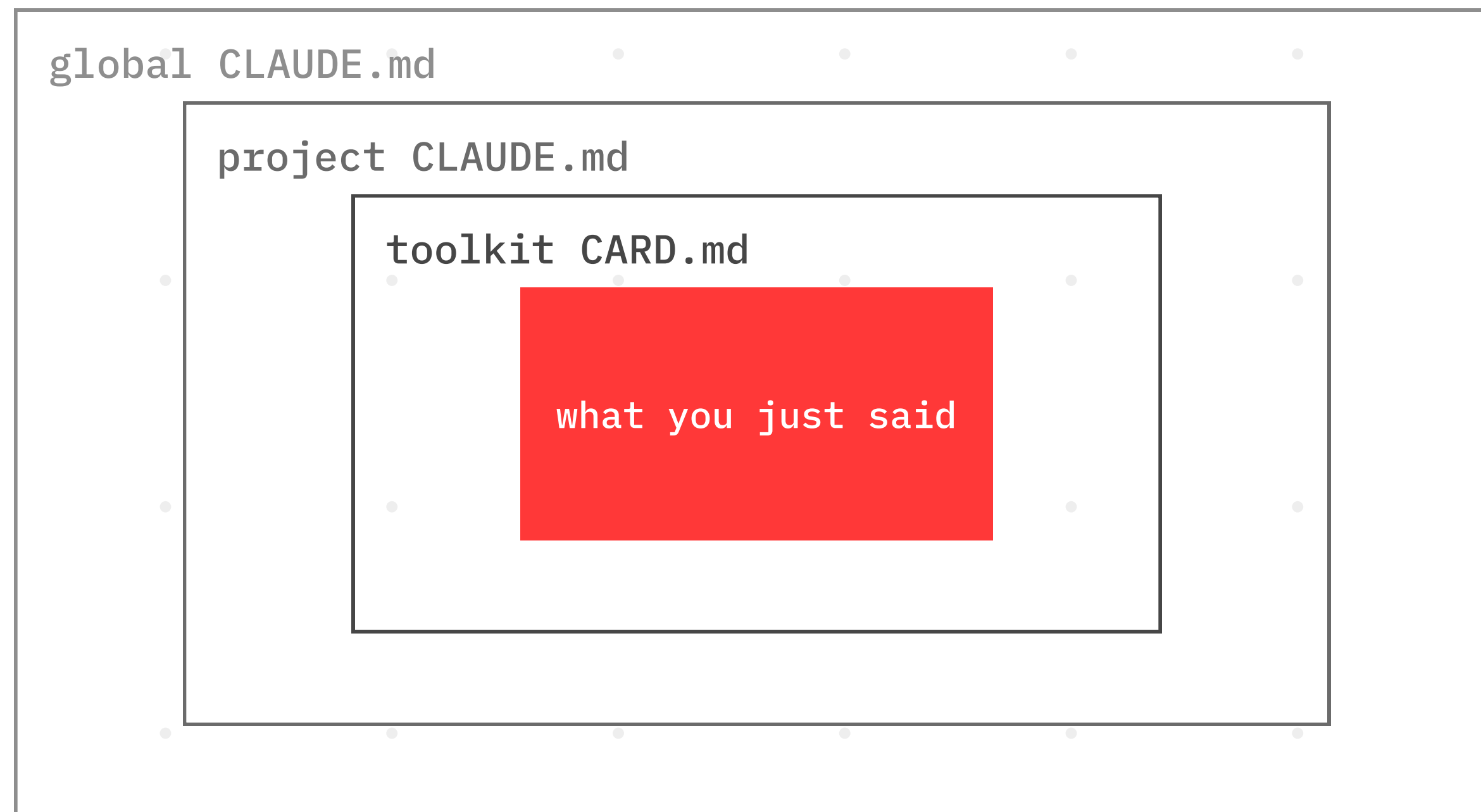
■ arrived with it

■ still on disk, not loaded

>_ memory --who-wins

When they disagree, the narrower one wins.

Four rings of instruction, and they do conflict. The one closest to the work overrides the one furthest from it.



■ global CLAUDE.md

How I work everywhere. Who you are, the cast, the toolkit, the house rules.

■ project CLAUDE.md

How I work in this folder. Overrides the global wherever the two differ.

■ toolkit CARD.md

How a tool actually behaves. Cards win on conflict, by your own house rule.

■ what you just said

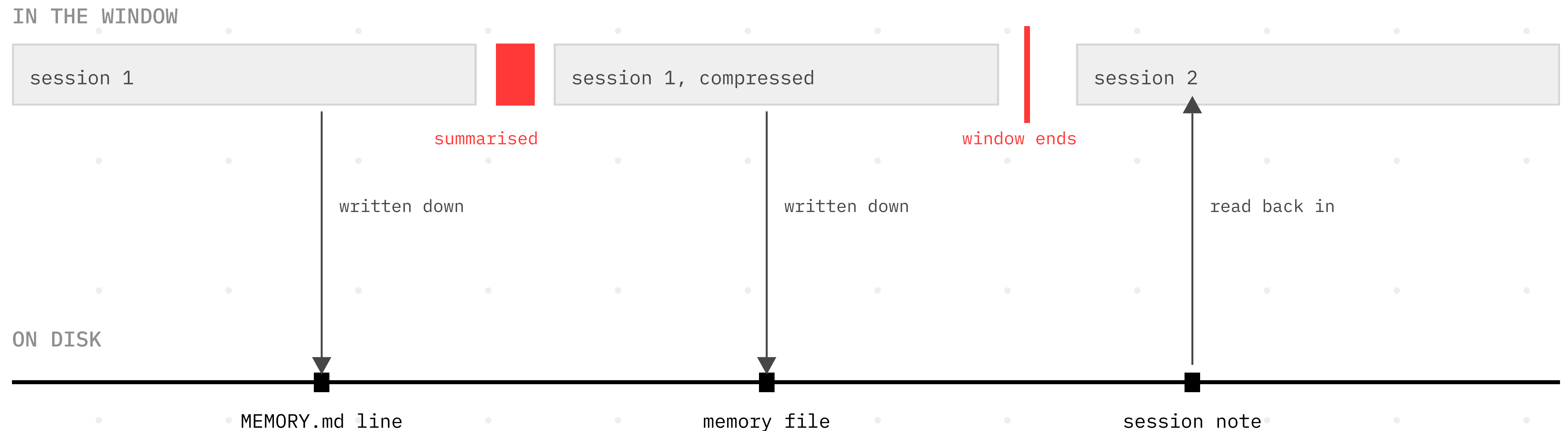
The live instruction. It beats every file on disk, including this one.

The rule is not seniority. It is proximity.

>_ memory --what-survives

The conversation is not the memory.

Context fills up, gets compressed, and eventually goes. Only what got written to a file is still here the next time you open a session.

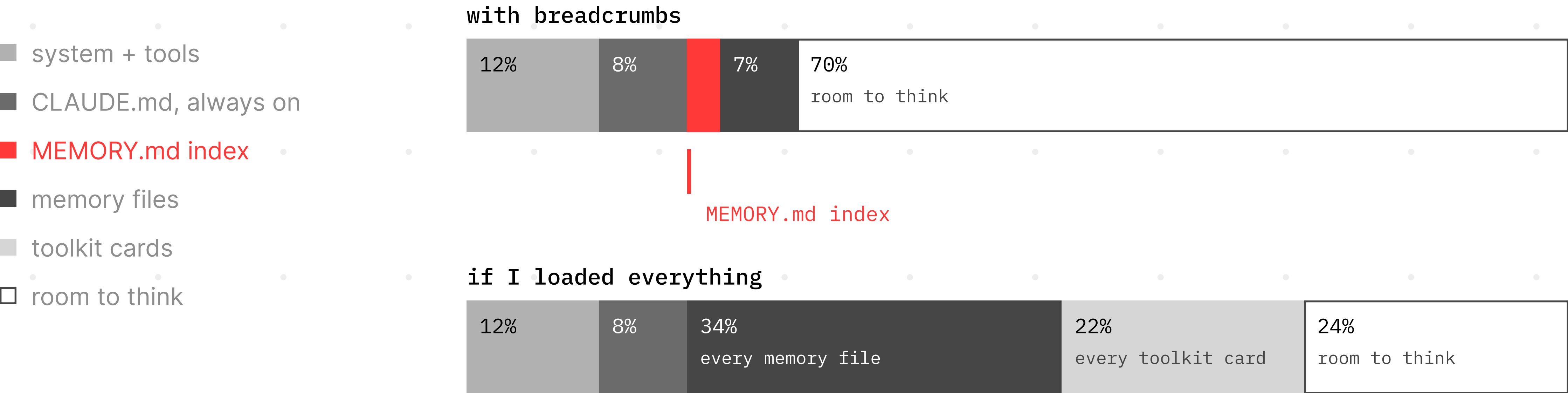


Everything else was only ever borrowed.

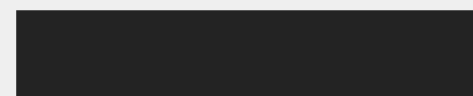
>_ memory --attention-budget

Everything loaded is something not thought about.

A session opens with a fixed plate. Breadcrumbs exist so the plate can hold pointers instead of documents, and the thinking gets what is left.



Three percent of the plate decides the other ninety-seven.



greg@superunknown.design

superunknown.