halvic labs

# Skill Audit: What Depreciates, What Appreciates

CHEATSHEET · VERSION: JULY 2026

**Your value moves from *producing code* to *owning systems*.** When code becomes cheap, what counts isn't how well you operate the tool — it's which of your abilities gain value and which lose it.

## The balance sheet

| Depreciates | Appreciates |
|---|---|
| **Typing boilerplate and CRUD from memory** — the agent is faster, every day | **Specifying, not prompting** — a precise, well-reasoned brief is the new core artifact |
| **Knowing syntax and API signatures by heart** — that was never competence, just repetition | **Judgment at volume** — distrusting generated code and convicting it |
| **Being the fastest typist on the team** — writing speed stopped being the bottleneck | **Thinking in systems, not files** — architecture, data flows, layer boundaries |
| **Framework trivia** — "what was that parameter called again?" is a solved question | **Translating between business and technology** — what should it do, for whom, how do you know it worked? |

None of these are new: it's classic software craft — freed from the typing that hid it for years.

## Self-audit

- ☐ Can I decompose a problem into a precise spec — inputs, edge cases, expected failure behaviour — before I prompt?
- ☐ Do I distrust generated code even at volume — and can I convict it, instead of waving every draft through?
- ☐ Do I think in systems, not files — seeing the architecture, data flows and layer boundaries where the agent only sees the local slice?
- ☐ Can I translate what the thing is supposed to do for the business, for whom — and how you know it worked?

## The experiment this week

No course, no certification — one experiment:

- ☐ Take your next task and write the spec **before** you fire up the agent.
- ☐ Capture everything before the first prompt: inputs, edge cases, expected failure behaviour — and the tests that must prove it.

- [ ] Then measure: how many corrective prompts did you save?
- [ ] Then measure: how much better was the first draft?

You'll notice the work that matters happens *before* and *after* the generation — that's exactly where the AI systems expert grows.

---

The full article: https://halvic.ch/en/blog/developer-skills-that-survive-ai · The deep dive: the audiobook *From Developer to AI Systems Expert* ("Vom Entwickler zum KI-System-Experten").

---

 