

Vorwort

Schon Mitte 2024 haben wir das erste Coding-CLI (*Command Line Interface*) ausprobiert. Das Programm *Aider* ließ erahnen, wie die Nutzung von KI-Tools im Terminal funktionieren könnte. Allerdings waren die Sprachmodelle damals noch nicht so gut wie heute. Auch *Aider* war in seinen Funktionen keineswegs auf dem Niveau von Claude Code. Kurz und gut: Es war ein interessantes Experiment, aber unsere Begeisterung hielt sich in Grenzen.

Im Februar 2025 stellte Anthropic die erste Version von Claude Code vor. Im Herbst 2025 erlebte das Programm seinen Durchbruch. Der Zeitpunkt ist kein Zufall: Kommerzielle Sprachmodelle erreichten erstmalig eine Qualität, die agentisches Programmieren ohne die Kontrolle jeder Zeile möglich machte.

Dennoch war uns das Potenzial des Programms nicht von Anfang an klar. Im Januar 2026 diskutierten wir noch darüber, ob wir der Neuauflage unseres Buchs »Coding mit KI« ein eigenes Kapitel über CLI-Tools spendieren sollten. (Das haben wir zum Glück getan, mit einem Fokus auf Claude Code.)

Neun Monate später ist für uns ein Arbeiten ohne Claude Code schlicht unvorstellbar. Wir nutzen Claude Code jeden Tag – und nicht nur zur Software-Entwicklung, sondern für eine Menge weiterer Dinge: administrative Arbeiten erledigen, Setup-Probleme lösen, Sicherheitsprobleme suchen, die Indexeinträge für dieses Buch abgleichen, Linux-Prüfungen korrigieren! Eigentlich vergeht keine Stunde, in der nicht auf einem unserer Rechner oder Server ein Terminal mit einer `claude`-Session offen ist.

»Coding is Solved«

»Coding is Solved« – dieses Zitat von Claude-Code-Erfinder Boris Cherny war Anfang 2026 (auch) ein flotter Werbeslogan für das eigene Produkt. Aber größtenteils stimmt die Aussage. KI-Tools programmieren 95 % des Codes schneller und besser als wir. Natürlich wollen und können wir Ihre Fähigkeiten hier nicht beurteilen, aber vermutlich gilt diese Aussage auch für Sie. Über die Quote von 95 % ließe sich diskutieren: Je nach Programmiersprache und Fachgebiet sind es manchmal nur 90 %, dann aber wieder 98 %.

Letztlich ist die Prozentzahl zweitrangig. Entscheidend ist: Einerseits programmieren KI-Tools mittlerweile *richtig* gut. Andererseits sind sie keineswegs perfekt und werden es wohl auch nie werden. Das Problem sind nicht Syntaxfehler oder Halluzinationen. Vielmehr passiert es, dass die KI nicht richtig versteht, was Sie wollen, dass sie sich in einer Sackgasse verirrt, alte Bibliotheken verwendet, Sicherheitsprobleme verursacht usw.

Menschliche Kontrolle bleibt also unumgänglich. Die größte Hürde bei der Software-Entwicklung hat sich vom Coding an sich zum Testen verlagert. Automatisierte Tests einzelner Funktionen sind der Idealfall, müssen aber von »echten« Tests des Gesamt-Setups begleitet sein, mit richtigen Daten, in der kompletten Benutzeroberfläche usw.

Wozu ein KI-Agent im Terminal?

Eigentlich ist es absurd: Jahrzehntelang wurden moderne Entwicklungsumgebungen optimiert. Cursor, IntelliJ, Visual Studio, VS Code, Xcode usw. kombinieren unendlich viele Features mit Bedienungskomfort und zuletzt auch mit KI-Funktionen. Dennoch zeigen immer mehr Entwickler diesen Programmen die kalte Schulter und verlagern ihre Arbeit in das schmucklose Terminal. Erstaunlicherweise handelt es sich dabei nicht um die Linux-Fraktion, die sich im Terminal sowieso zu Hause fühlt. Laut mehreren Umfragen arbeitet die Mehrheit der Claude-Code-Anwender unter macOS.

- Das Terminal ist die logische Konsequenz der These »Coding is Solved«. Sie wollen den Code im Regelfall gar nicht sehen. Oft interessiert Sie nicht einmal, welche Dateien Claude Code gerade verändert. Vielmehr geben Sie Anweisungen: »Plane diese neue Funktion! Implementiere sie! Teste dies! Korrigiere dort!«

Dazu reicht das Terminal. Eine Entwicklungsumgebung mit einem Dutzend Seitenleisten, Nebenfenstern und Einstellungsdialogen ist da nur im Weg. Die IDE verstellt den Blick auf das Wesentliche: den Dialog mit der KI, Ihrem virtuellen Coding-Partner.

- Dieser Dialog muss nicht zwangsläufig im Terminal erfolgen. Anthropic selbst bietet Claude Code auch außerhalb des Terminals an: in der Claude-Desktop-App, im Webbrowser und in Cloud-Sessions. (Bei Cloud-Sessions landet Ihr Code in einer virtuellen Maschine von Anthropic und wird dort ausgeführt und getestet.)

Der entscheidende Vorteil des Terminals besteht darin, dass alle auf Ihrem Rechner installierten Kommandos, CLIs und Programmierwerkzeuge direkt von Claude Code ausgeführt werden können. Die von diesen Kommandos gelieferten Ergebnisse werden direkt vom Sprachmodell übernommen und ausgewertet; bei Bedarf können sie im Terminal angezeigt werden. Der Arbeits- und Datenfluss ist geradlinig und ohne Umwege. (Die Desktop-App kann ebenfalls lokale Kommandos ausführen und bietet zudem eine grafische Diff-Ansicht. Allerdings fehlen ihr einige Funktionen der Terminal-Version, z. B. interaktive Slash-Kommandos, die direkte Ausführung von Shell-Kommandos mit `!` oder die Automatisierung mit `claude -p`.)

Claude Code im Terminal und moderne IDEs schließen sich keineswegs aus! Viele Developer verwenden beides: Auch wenn das Coding an sich im Terminal erfolgt, hilft die IDE dabei, die Gesamtstruktur des Projekts zu überblicken, einzelne Code-Dateien anzusehen und Tests durchzuführen.

Warum gerade Claude Code?

Agentische Coding-CLIs gibt es mittlerweile fast wie Sand am Meer. Was ist so besonders an Claude Code? Anthropic ist das Kunststück gelungen, schneller und agiler als die Konkurrenz zu erraten, was Programmiererinnen und Entwicklern wichtig ist. Dank KI-Unterstützung auch in der Weiterentwicklung von Claude Code werden neue Funktionen im Takt implementiert. Trotz aller terminalbedingten Einschränkungen ist die Textoberfläche äußerst intuitiv zu bedienen. *It just works!*

Wir haben im vergangenen Jahr diverse CLIs ausprobiert. Bisher hat uns kein Konkurrenzprodukt im gleichen Ausmaß überzeugen können. Natürlich werden andere CLIs langfristig aufholen, gleiche oder ähnliche Funktionen realisieren. Einerseits ist das gut: Sie brauchen keine Angst vor einem Lock-in-Effekt zu haben. Es *gibt* Alternativen (siehe dazu auch [Kapitel 10](#), »OpenRouter, lokale Modelle und Claude-Code-Alternativen«). Andererseits profitiert Anthropic davon, als Erste am Markt zu sein. Viele Entwickler machen sich nicht mehr die Mühe, Konkurrenzprodukte überhaupt auszuprobieren, weil sie schon ein Claude-Abo haben und zufrieden damit sind.

Das hohe Entwicklungstempo hat allerdings eine Kehrseite. Claude Code bietet mittlerweile eine kaum noch überschaubare Fülle von Konfigurations-, Erweiterungs- und Logging-Mechanismen. Geplant war dieses Buch mit 250 Seiten. Doch zwischen Konzept und Abgabe kamen laufend neue Funktionen hinzu. Außerdem stießen wir auf Features, die es schon länger gab, die wir aber schlicht nicht kannten. Wir sehen die Gefahr, dass sich Claude Code zu Bloatware entwickelt. Nicht jede neue Funktion ist ein Gewinn. Das KISS-Prinzip (*Keep it simple, stupid*) gilt auch für KI-Werkzeuge. Für Sie bedeutet das: Sie müssen längst nicht alles kennen. Viele Funktionen werden Sie nie brauchen. Wir konzentrieren uns in diesem Buch auf die Features, die sich in unserem Alltag bewährt haben.

Was bietet dieses Buch?

Dieses Buch ist kein Handbuch, das alle Optionen und Slash-Kommandos der Reihe nach abarbeitet. (Eine kompakte Referenz gibt es im letzten Kapitel trotzdem; Sie soll den Überblick geben, der in der umfangreichen Online-Dokumentation mitunter verloren geht.) Stattdessen verbinden wir die Grundlagen von Claude Code mit vielen Beispielen aus unserem Alltag. Das sind keine für das Buch konstruierten Demos, sondern echte Projekte: ein Partyspiel für die eigene MP3-Sammlung, eine über Jahre gewachsene PHP-Anwendung, ein Kundenprojekt in C#, die Backups eines Docker-Hosts, ein widerspenstiger OpenWrt-Router. Sogar dieses Buch dient uns mehrfach als Beispiel.

Nach einem kompakten Einstieg zeigen zwei umfangreiche Kapitel, wie sich die Arbeit mit Claude Code in der Praxis anfühlt: beim Start eines neuen Projekts und im Umgang mit bestehenden Codebasen. Die folgenden Kapitel vertiefen jeweils ein Thema, von Permissions über Skills und MCP bis hin zu Subagenten und Worktrees. Auch dort stehen

Beispiele im Mittelpunkt. Dass Claude Code weit mehr kann als programmieren, zeigen wir anhand der Server-Administration. Den Abschluss bilden Alternativen zum Anthropic-Abo, Tipps aus unserem Alltag und eine kurze Referenz der wichtigsten Slash-Kommandos, Claude-Code-Optionen und Umgebungsvariablen.

Wir zeigen nicht nur, was funktioniert. Mindestens ebenso lehrreich sind die Momente, in denen Claude Code danebenliegt: wenn es eine Prüfung eigenmächtig in eine Reparatur verwandelt, eine plausible, aber falsche Erklärung erfindet oder sich festfährt. Aus solchen Episoden leiten wir die Arbeitsweisen ab, die sich bei uns bewährt haben. Für viele Aufgaben bietet Claude Code mehrere Mechanismen an, etwa `CLAUDE.md`, Skills, MCP-Server oder Hooks. Wir sagen klar, welche wir bevorzugen und warum. Dieses Wissen entsteht erst durch monatelange tägliche Nutzung.

Als wir das Konzept für dieses Buch entwickelt haben, wollten wir uns zu 100 % auf Claude Code konzentrieren. Diesem Ansatz sind wir treu geblieben: Auf die Erläuterung von KI-Grundlagen verzichteten wir. Andere Coding-CLIs kommen nur in einem kurzen Ausblick in Kapitel 10 vor. Andere KI-Werkzeuge und Entwicklungsumgebungen mit KI-Funktionen sind kein Thema. Auch die Claude-Desktop-App, Claude im Webbrowser und Cloud-Sessions lassen wir weitgehend weg. Einzige Ausnahme ist Abschnitt 11.2: Dort beschreiben wir, wie Claude Code mit dem Smartphone, mit IDEs und mit anderen Anthropic-Werkzeugen zusammenspielt.

Viel Spaß!

Wenn Sie dieses Buch in Händen halten, kennen Sie Claude Code vermutlich schon ein wenig oder haben von Ihren Freunden gehört, die es bereits einsetzen. Im ersten Fall sind wir uns sicher, dass Sie in diesem Buch – gerade anhand der Beispiele – eine Menge dazulernen können. Im zweiten Fall warten noch mehr Aha-Erlebnisse auf Sie.

Im letzten Jahrzehnt gab es wenige Produkte oder Programme, die uns so bedingungslos überzeugt haben wie Claude Code. Lassen Sie sich von unserer Begeisterung für dieses neue Konzept des Codings anstecken!

Michael Kofler (<https://kofler.info>)

Bernd Öggl (<https://webman.at>)