CHEATSHEET · CLAUDE ACADEMY

CLAUDE.md
5 Vorlagen

Bewährte Startpunkte für die wichtigsten Projektarten — zum Kopieren, Anpassen, direkt commiten. Halte deine CLAUDE.md kurz und konkret.

tbai
Stand Mai 2026

01 · Universal

Minimal — der Rumpf für jedes Projekt

Wenn du nicht weißt, was rein soll: dieser Rumpf reicht für die erste Session. Erweitern, sobald Wiederholungen auftauchen.

```
# Projektname

## Stack
- [Sprache] [Version], [Framework]

## Befehle
- `[run]` – startet die App
- `[test]` – führt Tests aus

## Konventionen
- [1-2 Regeln, z.B. "TypeScript strict, keine any"]
- [Style-Note, z.B. "Imports nach Pfad-Tiefe sortieren"]
```

02 · Web-App

Next.js / Astro / SvelteKit

```
# [Projektname] – Web-App

## Stack
- Next.js 15, TypeScript strict, Tailwind 4
- DB: Postgres via Prisma
- Auth: NextAuth.js

## Befehle
- `pnpm dev` – Dev-Server (Port 3000)
- `pnpm build && pnpm start` – Production
- `pnpm test` – Vitest
- `pnpm db:migrate` – Prisma-Migration

## Konventionen
- Server-Components-First. `use client` nur wenn nötig.
- Form-Validierung mit Zod-Schemas, geteilt zwischen Client und Server.
- Tests neben dem File: `foo.ts` ↔ `foo.test.ts`.

## Done-Definition
- Tests grün, Typecheck grün, kein Lint-Warning.
- Bei DB-Änderung: Migration + Seed-Update.
```

03 · Python-Tool

CLI & Data-Tools

```
# [Tool-Name]

## Stack
- Python 3.12, uv für Dependencies
- Entry-Point: `python -m ` oder via `pyproject.toml` script

## Befehle
- `uv sync` — Dependencies aktualisieren
- `uv run pytest` — Tests
- `uv run ruff check && uv run ruff format` — Lint + Format
- `uv run mypy .` — Type-Checking

## Konventionen
- Type-Hints überall. `from __future__ import annotations`.
- Errors: spezifische Exceptions, keine `except Exception`.
- Logging über `logging`-Modul, kein `print`.

## Wichtig
- Keine `.env`-Variablen committen. `.env.example` pflegen.
```

04 · Monorepo

Hierarchische CLAUDE.md

Im Monorepo: Root-CLAUDE.md mit Shared-Rules + eine CLAUDE.md pro Paket mit Paket-spezifischen Regeln. Claude liest beide.

```
# Root — gemeinsame Regeln

## Stack
- pnpm Workspaces, TypeScript, Turbo

## Pakete
- `apps/web` — Next.js (eigene CLAUDE.md)
- `apps/api` — Hono (eigene CLAUDE.md)
- `packages/ui` — Shared Components
- `packages/db` — Prisma Schema (Single Source of Truth)

## Befehle (Root)
- `pnpm dev` — alle Apps parallel
- `pnpm test --filter=` — gezielt testen

## Konventionen
- Schema-Änderungen IMMER zentral in `packages/db`.
- Cross-Package-Imports nur via Workspace-Paths (`@org/ui`).
```

05 · Buchprojekt

Solo-Author · Markdown → PDF/EPUB

```
# [Buchtitel]

## Format
- Amazon KDP (Print + Kindle/EPUB)
- Ein .md pro Kapitel in `manuscript/`
- Reihenfolge in `metadata.json`

## Sprache & Stil
- Deutsch, formal (Sie-Form/neutral)
- KI/Tech-Begriffe auf Englisch
- Keine Emojis im Buchtext

## Build
- `python build.py` — kompiliert Markdown → HTML → PDF (Chrome) + EPUB
- Output nach `build/`

## Wichtig
- Kapitel-Reihenfolge aus `metadata.json` respektieren
- HTML/CSS-Vorlage nicht ohne explizite Anweisung ändern
- Manuskript-Stand vor Build commiten
```