Anthropic Professor-Verslag — Sjakie's kennisbank

Doel: alles wat ik moet weten over Claude Code, de Claude API en Anthropic-producten op één plek, zodat ik als "professor" kan werken en je proactief kan adviseren.
Aangemaakt: 2026-06-03 · **Door:** Sjakie (educatie-opdracht van Jamal)
Onderhoud: bij elke nieuwe release deze file updaten (zie deel 6 — release-monitoring).

---

0. TL;DR — de 10 dingen die er het meest toe doen

1. **Opus 4.8 is de huidige flagship** (`claude-opus-4-8`): 1M context, 128k output, $5/$25 per MTok, adaptive-thinking-only, kennisgrens jan 2026. Sonnet 4.6 (`claude-sonnet-4-6`) en Haiku 4.5 (`claude-haiku-4-5`) eronder.
2. **Skills > losse prompts** — herhaalbare workflows horen in een SKILL.md, niet steeds opnieuw uitgetypt.
3. **Subagents (Task/Agent tool) voor parallel werk** — beschermt hoofd-context, doet onafhankelijke taken tegelijk.
4. **Hooks = automatisering op events** — bijv. mijn Telegram-reply-enforcer is een Stop-hook.
5. **CLAUDE.md-hiërarchie** bepaalt mijn gedrag; user-instructies > skills > default systeemprompt.
6. **Auto-memory** = mijn lange-termijn geheugen tussen sessies (deze map).
7. **Prompt engineering kern:** wees specifiek, geef voorbeelden, laat me redeneren (thinking), structureer met XML/secties.
8. **`effort` -parameter** stuurt diepte van redeneren (low→max; xhigh voor code).
9. **Release-bronnen monitoren** = GitHub releases API + platform release-notes + anthropic.com/news + status-page.
0. **Context is kostbaar** — chunk grote taken, schrijf naar files, vat samen in chat.

1. Claude Code — expert-referentie

Kernconcepten

- **CLAUDE.md** — projectinstructies, automatisch geladen. Hiërarchie: globaal (`~/.claude/CLAUDE.md`) → project → submap. Overschrijft default gedrag.
- **Settings** (`settings.json`): permissions (allow/deny/ask), MCP-servers, hooks, model-keuze. Precedentie: lokaal project > project > user > systeem.
- **Permission modes:** default (vraagt), acceptEdits, plan, bypassPermissions.

Skills

- Map met `SKILL.md` (+ frontmatter `name` , `description`). Beschrijving bepaalt wanneer ik 'm automatisch inzet.
- Mijn skills: beeldgeneratie-gate, end-session, higgsfield-content-factory, seo-brief, start, telegram-doctor, ui-ux-pro-max.
- Best practice: korte, scherpe `description` met trigger-woorden; rigide vs flexibele skills.

Subagents

- Via Task/Agent tool. Eigen context-window, geven beknopte samenvatting terug.
- Types: Explore (read-only zoeken), general-purpose, en gespecialiseerde reviewers.
- Gebruik voor: parallelle research, grote codebase-verkenning, context-bescherming.

Hooks (event-gedreven shell-commando's)

- 15+ events: PreToolUse, PostToolUse, Stop, SubagentStop, UserPromptSubmit, SessionStart, PreCompact, Notification, etc.
- Mijn actieve hooks: Telegram-reply-enforcer (Stop), queue-hook, SessionStart-compact.
- Use cases: validatie vóór tool-call, auto-format, smoke-tests, enforce conventies.

MCP (Model Context Protocol)

- Externe tools/data via servers. Transports: stdio, SSE, HTTP.

- Mijn MCPs o.a.: telegram, higgsfield, fal-ai, github (uit — zelden nodig), playwright, context7, computer-use, plaud.
- `claude mcp list/add/remove` . Bij Telegram: `--channels` flag + `--transport sse` essentieel.

Memory & sessie

- Auto-memory map = persistente kennis tussen sessies (MEMORY.md index + losse .md's).
- Plugins, IDE-integraties (VS Code/JetBrains), Agent SDK voor eigen agents.

Top valkuilen

- Output-token-limiet → chunk + schrijf naar file.
- Context vol → samenvatten, subagents inzetten.
- Te veel bash-probes bij debug → lees liever config/logs.

---

2. Claude API & platform

Modellen (live, juni 2026)

MODEL	ID	CONTEXT	OUTPUT	PRIJS IN/OUT	KENMERK
Opus 4.8	`claude-opus-4-8`	1M	128k	$5/$25 MTok	flagship, adaptive-thinking-only, jan-2026 cutoff
Sonnet 4.6	`claude-sonnet-4-6`	groot	hoog	midden	balans snelheid/kwaliteit
Haiku 4.5	`claude-haiku-4-5`	—	—	goedkoop	snel, subagents/QC

Belangrijke API-features

- **Extended/adaptive thinking** — model redeneert vóór antwoord; `effort` (low→max, xhigh voor code) stuurt diepte.

- **Tool use / function calling** — gestructureerde tool-aanroepen (zoals ik nu doe).
- **Vision** — beelden analyseren (basis van mijn visual_qc.py + generation_qc.py).
- **Prompt caching** — herhaalde context goedkoper/sneller (5-min TTL).
- **Batch API** — async bulk, goedkoper.
- **Citations, PDF-support, structured output.**
- **Let op:** prefill gedeprecieerd op 4.6+; sampling-params capped op 4.7/4.8.

Prompt engineering — kern

1. Wees expliciet en specifiek (geen ruimte voor gokken).
2. Geef voorbeelden (few-shot).
3. Laat me redeneren (thinking) bij complexe taken.
4. Structureer met XML-tags / duidelijke secties.
5. Rol/persona toewijzen.
6. Taken opdelen (chaining) bij meerstaps.
7. Output-format expliciet voorschrijven.

---

3. Anthropic Academy & cursussen

- **anthropic.skilljar.com** — officiële Academy (gratis cursussen).
- **github.com/anthropics/courses** — o.a. prompt engineering, real-world prompting, tool use, model context protocol.
- **github.com/anthropics/anthropic-cookbook** — praktische code-recepten (vision, tool use, RAG, embeddings, etc.).
- Kernlessen: prompt-discipline, tool use orchestratie, evaluaties bouwen, RAG-patronen.

---

4. Documentatie-bronnen (bookmarks)

- **platform.claude.com/docs** — hoofddocumentatie (verhuisd van docs.anthropic.com).

- **docs.claude.com/en/docs/claude-code** — Claude Code docs.
- **anthropic.com/news** — aankondigingen/blog.
- **status.claude.com** — service-status.
- **github.com/anthropics** — courses, cookbook, SDK's, claude-code.

---

5. Hoe ik dit toepas op ons werk

- **Beeldgeneratie:** vision-QC loop (generation_qc.py) + 1-testfoto-gate = minder correctierondes.
- **Parallel werk:** subagents voor research/generatie tegelijk (Pilarczyk-stijl).
- **Context-discipline:** grote output → file + 5-regel samenvatting in Telegram.
- **Model-routing:** Opus 4.8 standaard, Haiku voor goedkope QC/subagents, Ollama voor triviaal.
- **Skills uitbreiden** waar we workflows herhalen.

---

6. Release-monitoring — hoe ik op de hoogte blijf

Bronnen (machine-leesbaar eerst):
1. **GitHub releases API/Atom** — `github.com/anthropics/claude-code/releases` (+ SDK-repos). Meest betrouwbaar, gestructureerd.
2. **platform/Claude Code release-notes** — scrapen (docs.claude.com release notes).
3. **anthropic.com/news** — geen RSS, scrapen.
4. **status.claude.com** — JSON/Atom voor incidenten.

Ontwerp (voorstel, nog niet gebouwd): dagelijkse check → diff tegen state-file → bij nieuwe release een korte Telegram-alert naar Jamal met wat het is + wat het voor ons betekent.

Laatste update: 2026-06-03. Update deze file bij elke relevante Anthropic-release.