

Les 3 — Lesstof

Cursor — AI-Powered Code Editor

Vak: AI Fundamentals
Cursus: AI Developer
Volgende les: Les 4 — TypeScript

Vak: AI Fundamentals **Vorige les:** Les 2 — OpenCode **Volgende les:** Les 4 — TypeScript Fundamentals

Inhoud

1. [Wat is Cursor](#1-wat-is-cursor)
2. [Setup + eerste keer opstarten](#2-setup--eerste-keer-opstarten)
3. [De vier kern-features](#3-de-vier-kern-features)
4. [Chat (Cmd+L)](#4-chat-cmdl)
5. [Inline Edit (Cmd+K)](#5-inline-edit-cmdk)
6. [Composer / Agent (Cmd+I)](#6-composer--agent-cmdi)
7. [Tab — predictive completion](#7-tab--predictive-completion)
8. [@-mentions — context erbij halen](#8--mentions--context-erbij-halen)
9. [.cursorrules — project-context](#9-cursorrules--project-context)
10. [Debug-flow met AI](#10-debug-flow-met-ai)
11. [Wanneer Cursor vs Console](#11-wanneer-cursor-vs-console)

1. Wat is Cursor

Cursor is een AI-first code editor — een fork van VS Code met AI-features verweven in elk onderdeel. Geen plugin, geen sidebar — AI is overall: tab-completion, chat, inline edit, agent mode.

Waarom Cursor?

- Vertrouwde VS Code interface (alle extensies werken)
- AI features die écht in je workflow zitten
- Multi-file edits met diff-preview

- Background Agents (later in de leerlijn — Les 15)
- Werkt met OpenAI, Anthropic, of eigen API keys

Cost

- **Hobby (gratis)** — limited, Tab completion, basic chat
- **Pro (\$20/maand)** — alles, onbeperkt, Background Agents
- **Business** — teams, SSO, audit

Voor deze leerlijn: Pro aanrader. Hobby werkt voor leren maar je raakt rate-limits.

2. Setup + eerste keer opstarten

Installatie

Download van cursor.com — installeer net als VS Code.

Eerste opstarten

1. Sign in met email of GitHub
2. Cursor vraagt: "Import VS Code settings?" — Ja
3. Kies thema, snelkoppelingen
4. Opt-in op AI training of opt-out (privacy keuze)

Model kiezen

Settings → Models. Defaults:

- Cursor Auto (default — Cursor kiest beste model)
- Claude Sonnet 4.6 (default voor code)
- GPT-5 (alternatief)
- o3 (reasoning — trager maar slimmer)

Vuistregel: Sonnet voor 90%, o3 voor complex.

Een project openen

Twee opties:

- File → Open Folder (zoals VS Code)
- Terminal: `cursor .` in je project-folder

3. De vier kern-features

Cursor heeft veel features. Vier die je dagelijks gebruikt:

Feature	Shortcut	Wanneer
Tab	(auto)	Voltooi-suggesties tijdens typen
Inline Edit	Cmd+K	Korte edit op selectie of cursor-positie
Chat	Cmd+L	Vragen stellen, code uitleggen
Composer/Agent	Cmd+I	Multi-file edits, refactors

Deze leren = 80% van Cursor's waarde gebruiken.

4. Chat (Cmd+L)

Wat: sidebar-chat. Type vraag, krijg antwoord. Net als ChatGPT maar mét je code als context.

Wanneer

- Code uitleggen ("wat doet deze functie?")
- Algemene vragen tijdens coden ("hoe gebruik je useMemo?")
- Debug-hulp ("waarom werkt dit niet?")

Context erbij

- @filename — voeg specifieke file toe als context
- @Codebase — hele repo (gebruikt embeddings)
- @Web — search het web
- @Docs — geïmporteerde documentatie
- Sleep een file naar chat — toegevoegd als context

Agent-mode binnen Chat

Switch knop bovenaan: Ask | Agent. Met Agent mode kan Chat ook files bewerken — vergelijkbaar met Composer.

5. Inline Edit (Cmd+K)

Wat: edit-popup op je cursor-positie. Beschrijf wat je wilt, AI past selectie of nieuwe code aan.

Voorbeelden

Selecteer een functie + Cmd+K:

"Voeg error handling toe"

AI past selectie aan, diff zichtbaar, accept/reject.

Lege regel + Cmd+K:

"Genereer een TypeScript interface voor een User met name, email, role"

AI inserts code.

Wanneer

- Snelle wijziging op één plek
- Code generaten tussen bestaande regels
- Selectie aanpassen

Voor multi-file edits → Composer.

6. Composer / Agent (Cmd+I)

Wat: de zwaarste AI-feature. Multi-file edits met preview van alle diffs.

Voorbeeld

Cmd+I opent Composer panel:

"Voeg een /about pagina toe met team-leden in een grid. Match styling van homepage. Gebruik Tailwind. Maak een aparte component voor team-member."

Composer:

1. Leest app/page.tsx voor stijl
2. Maakt app/about/page.tsx
3. Maakt components/TeamMember.tsx
4. Toont alle diffs

Jij accepteert per file of "Accept All".

Wanneer

- Nieuwe features (multi-file)
- Refactors
- Bugfixes die meerdere files raken

Terminal commands

Composer kan ook terminal-commands runnen — vraagt approval. "Run pnpm install" → Composer doet het.

7. Tab — predictive completion

Cursor's Tab-completion ziet **wat je gaat typen voordat je 't typt**. Vergelijkbaar met GitHub Copilot maar slimmer (gebruikt je hele file + recent edits).

Hoe gebruiken

- Type — suggesties verschijnen grijs
- Tab — accepteer
- Esc — afwijzen

Cursor Tab vs Copilot

Cursor Tab kan:

- Multi-line suggesties (hele functie ineens)
- Cursor verplaatsen naar volgende edit-locatie (handig bij refactors)
- Suggesties op basis van wat je net hebt geschreven elders

Verschil voelbaar na een dag gebruik.

Wanneer uitzetten

Soms ga je sneller zonder. Cmd+Shift+P → "Cursor Tab" → toggle.

8. @-mentions — context erbij halen

@ in chat of Composer = context kiezen.

Soorten

Mention	Wat	Voorbeeld
@File	Specifieke file	@app/page.tsx
@Folder	Hele folder	@components/
@Codebase	Repo via embeddings	@Codebase
@Docs	Geïmporteerde docs	@Next.js
@Web	Web search	@Web "Next.js 16 caching"
@Git	Recent commits	@Git
@Definitions	Symbols	@Definitions:UserController

Goede prompts

Zonder mentions: AI gokt welke files relevant zijn (langzamer, soms fout). Met mentions: AI heeft exact wat 't nodig heeft.

```
@app/api/auth/route.ts @lib/supabase.ts
```

Voeg een endpoint toe /api/user dat de logged-in user returnt.
Gebruik supabase auth.getUser().

9. .cursorrules — project-context

.cursorrules = markdown-file in project-root die Cursor leest bij elke prompt. Net als AGENTS.md in Les 2.

Voorbeeld

```
# Cursor Rules — QuickPoll

## Stack
- Next.js 16 App Router
- TypeScript strict mode
- Tailwind v4
- Supabase (DB + Auth)

## Conventies
- Server Components default, "use client" alleen waar nodig
- Tailwind classes inline, geen CSS-files
- Geen comments tenzij gevraagd
- Volg bestaande naming (camelCase functions, PascalCase components)
- Throw geen errors uit RSCs — return { error } object

## Verboden
- Gebruik geen Next.js Pages Router (alleen App Router)
- Geen any-types in TypeScript
- Geen inline styles
```

Effect

AI checkt deze rules bij elke prompt. "Maak een Server Component" wordt automatisch "use client"-vrij.
"Voeg comment toe" wordt overgeslagen.

Beperkingen

- Cursor's parsing is niet 100% strikt — soms vergeet AI rules
- Hou 'm tussen 30-200 regels (te lang = vergeten)
- Test of rules echt werken — soms moet je expliciet refereren

10. Debug-flow met AI

Pattern 1 — Console error

Console error → kopieer → in chat plakken:

```
TypeError: Cannot read property 'name' of undefined  
at UserCard (components/UserCard.tsx:12)
```

AI: leest UserCard.tsx, vindt waar name undefined kan zijn, suggereert fix.

Pattern 2 — Onverwachte output

```
Mijn fetchData() returnt undefined ipv users. Hier is de code:  
  
@lib/api.ts  
  
Wat gaat er mis?
```

AI: leest code, identificeert (vaak) async/await issue of missende return.

Pattern 3 — Tests falen

Run test → copy output → chat:

```
Test "should return user list" faalt:  
Expected: Array(3)  
Received: undefined  
  
@__tests__/api.test.ts  
@lib/api.ts
```

AI: traceert, fixt.

Pattern 4 — Refactor

```
Refactor @app/api/users/route.ts naar gebruik van Supabase  
in plaats van mock-data.  
  
Gebruik @lib/supabase.ts voor de client.
```

Composer doet multi-file edit.

11. Wanneer Cursor vs Console

Cursor is geweldig maar niet altijd het antwoord.

Cursor wel

- Code schrijven, refactoren, debuggen
- Nieuwe features

-
- Boilerplate
 - Documentatie

Cursor niet

- Git workflow (gebruik gewoon git)
- File-management buiten project (Finder/Explorer)
- Terminal-heavy taken (gewone shell)
- Browser debuggen (DevTools)
- Een quick lookup (ChatGPT/browser)

Hybride

Veel pro devs: **Cursor voor IDE-werk, Claude.ai voor brainstormen, Terminal voor git.** Gebruik niet één tool voor alles.

Bronnen

- **Cursor docs:** <https://docs.cursor.com>
- **Cursor Tab features:** <https://docs.cursor.com/tab>
- **Cursor Composer:** <https://docs.cursor.com/composer>
- **.cursorrules examples:** <https://github.com/PatrickJS/awesome-cursorrules>
- **Cursor changelog:** <https://www.cursor.com/changelog>
- **Cursor forum (community):** <https://forum.cursor.com>