



Les 3

Cursor Basics

AI-first code editor - VS Code met superkrachten



Terugblik

Les 2 - OpenCode

CLI

AI in terminal naast je code

Plan vs Build

Veilig refactor of snel doen

AGENTS.md

Project-context in markdown

Git workflow

Push -> Vercel deploy

OpenCode = krachtig in terminal, maar...

- Geen visuele editor
- Geen multi-file diff preview ingebouwd
- Geen IDE features (jump-to-def, find references)

Vandaag: Cursor - alles in een editor + AI overal verweven.

Planning

Vandaag - 180 minuten

Welkom + terugblik 10 min

Wat is Cursor + setup 15 min

De vier kern-features 15 min

LIVE DEMO 1 - Chat + Inline Edit 15 min

LIVE DEMO 2 - Composer + @-mentions 15 min

Pauze 15 min

Cursorrules + debug-flow 15 min

Lesopdracht: debug-challenge 70 min

Afsluiting + huiswerk 10 min

Theorie 1

Wat is Cursor?

AI-first code editor. Fork van VS Code. AI verweven in elk onderdeel:

- v Vertrouwde VS Code interface - extensies werken
- v AI features die in workflow zitten, geen sidebar afterthought
- v Multi-file edits met diff-preview
- v Background Agents (Les 15)
- v Werkt met OpenAI, Anthropic, of eigen API keys

Cost:

Hobby	Gratis, limited - Tab + basic chat
Pro	\$20/mnd - alles, onbeperkt, Background Agents
Student	Gratis Pro - check cursor.com/students

Setup

Installeren + opstarten

- 1 Download van cursor.com (Mac/Windows/Linux)
- 2 Sign in met email of GitHub
- 3 Import VS Code settings? -> Ja
- 4 Settings -> Models -> Sonnet (default voor 90% taken)
- 5 File -> Open Folder, of in terminal: cursor .

Model-keuze:



Sonnet 4.6

default voor code - 90% taken



GPT-5

alternatief, goede taal-uitleg



o3

reasoning - trager maar slimmer (complex)

Theorie 2

De vier kern-features

Tab

(auto)

Predictive completion tijdens typen

Inline Edit

Cmd+K

Korte edit op selectie of cursor

Chat

Cmd+L

Vragen, uitleg, debug-hulp

Composer

Cmd+I

Multi-file edits, refactors

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

LIVE DEMO 1

Chat + Inline Edit - ~15 min

Chat (Cmd+L) - vragen, uitleg, refactor-suggestie

```
@app/page.tsx
```

```
Wat doet deze component? Refactor naar Server Component  
en haal data via Supabase op.
```

Inline Edit (Cmd+K) - selecteer functie, type prompt

```
// Selecteer functie + Cmd+K:  
> Voeg error handling toe + return Result<T> object  
  
// Diff zichtbaar, accept/reject per regel.
```

Tip: Inline Edit ook op een lege regel = code genereren op die plek.

LIVE DEMO 2

Composer + @-mentions - ~15 min

Composer (Cmd+I) = multi-file feature. Preview alle diffs voor accept.

```
@app/page.tsx @components/
```

Voeg een /about pagina toe met een team-grid.
Match styling van homepage. Gebruik Tailwind.
Maak een aparte TeamMember component.

@-mentions = context kiezen:

@File	specifieke file
@Folder	hele folder
@Codebase	hele repo via embeddings
@Docs	geïmporteerde documentatie
@Web	live web search

Theorie 3

Tab - predictive completion

Cursor ziet wat je gaat typen voordat je 't typt. Slimmer dan Copilot.

Hoe gebruiken

- v Type code, suggesties verschijnen grijs
- v Tab = accepteer suggestie
- v Esc = afwijzen
- v Cursor verplaatst zelf naar next edit

Cursor Tab kan...

- Multi-line suggesties (hele functie)
- Refactor follow-through
- Naam-renames in een file overnemen
- Imports automatisch toevoegen

Workflow-tip:

Type vlot door - laat Tab je completen. Verschil voelbaar na een dag.

Soms ga je sneller zonder - toggle via Cmd+Shift+P -> Cursor Tab.



Pauze

15 minuten

Theorie 4

.cursorrules - project-context

Markdown in repo-root. Cursor leest deze bij elke prompt.

```
# Cursor Rules - QuickPoll

## Stack
- Next.js 16 App Router, TypeScript strict, Tailwind v4, Supabase

## Conventies
- Server Components default, "use client" alleen waar nodig
- Tailwind classes inline, geen CSS-files
- Geen comments tenzij gevraagd
- camelCase functions, PascalCase components

## Verboden
- Geen any-types
- Geen Pages Router (alleen App Router)
```

Hou 'm tussen 30-200 regels. Te lang = AI vergeet de helft.

Theorie 5

Debug-flow met AI



Console error

Kopieer error -> chat. AI leest file, vindt + fixt.



Onverwachte output

@file plakken. AI traceert (vaak async issue).



Tests falen

Test output + @testfile -> chat. AI fixt.



Refactor

Composer + meerdere @file. Multi-file edit.

Theorie 6

Wanneer Cursor vs niet

Cursor wel

- v Code schrijven + refactoren
- v Debug-flows met context
- v Nieuwe features (multi-file)
- v Boilerplate + documentatie

Cursor niet

- x Git workflow - gebruik git CLI
- x Terminal-heavy taken - gewoon shell
- x File-management - Finder/Explorer
- x Browser debuggen - DevTools

Hybride aanpak in praktijk:

Cursor voor IDE-werk + Claude.ai voor brainstormen + Terminal voor git

Geen tool die alles wint. Pak per taak de beste.

LESOPDRACHT

Debug-Challenge (~70 min)

Drie levels:

Normaal

5 bugs in een React app

Hard

10 bugs in een Next.js app

Super Hard

Onbekend - jij ontdekt wat er stuk is

Aanpak:

- Unzip de starter, open in Cursor
- Gebruik Chat/Inline/Composer - elk feature minimaal 1x
- Maak een .cursorrules voor de stack
- Document elke bug + fix in een commit message

Inleveren: zip van fix + commit log via Teams.

Huiswerk

Voor Les 4 (TypeScript)

1. Refactor je v0-website met Cursor

Composer: voeg sectie + component toe. Volg .cursorrules.

2. Schrijf 5 prompts in je log

Wat vroeg je, wat kreeg je, was 't goed?

3. Optioneel: Debug-Challenge Hard afmaken

Als je ze in de les niet allemaal hebt gehaald

4. Inleveren via Teams

GitHub repo + prompt-log + reflectie

Tip: speel met @-mentions. Verschil tussen 'fix het' en '@file fix bug X'.

Volgende les

Les 4 - TypeScript Fundamentals

Eerste stap richting productie-code:

- Types, interfaces, generics
- Union + literal types
- Type narrowing + guards
- Hands-on: TypeScript Escaperoom (10 kamers)

Cursor + TypeScript = magie.

AI-suggesties worden veel beter zodra types in spel zijn.

Vragen?

Tot volgende week!