

Les 3 — Docenttekst (Autocue)

Fysieke les — letterlijk voorleesbaar. Slide-nummers tussen [SLIDE N].

Structuur: ±1,5 uur theorie/demo → kwartier pauze → ±1,25 uur hands-on

>

Voorbereiding — accounts en tools: - Cursor zelf geïnstalleerd en met Pro Student account ingelogd - Twee test-projecten klaar (één leeg, één met Hero al gebouwd als backup) - Browser tabs: cursor.com/students, cursor.com/download, Next.js docs - Terminal open in `~/web` voor de live demo

>

Voorbereiding — content: - `.cursorrules` snippet copy-paste klaar - Hero-prompt copy-paste klaar - Cmd+K tweak-prompt copy-paste klaar

>

Voorbereiding — backup: - Cursor Pro Student verificatie kan haperen — heb een tweede gedragsslijn klaar als studenten niet kunnen activeren - Screenshots van Hero-resultaat als live demo niet werkt

>

Voor uitgebreide handleiding zie `Les03-Docenttekst.md` (normale versie).

BLOK 1 — Welkom + Wat is Cursor

[SLIDE 1]

Welkom bij Les 3. Vandaag installeren we Cursor — de AI-powered editor die we vanaf nu de hele opleiding gaan gebruiken. Aan het eind van vandaag heeft iedereen Cursor draaien, het Student Plan actief, en jullie eigen Next.js project waarin je drie kleine componenten hebt gebouwd.

[SLIDE 2]

Drie dingen die je vandaag leert: Cursor installeren plus activeren met het Student Plan. De drie hoofdsneltoetsen — Tab, Cmd+K, Cmd+L. En een werkend Next.js-project waarin je een Hero-component hebt gebouwd.

De werkwijze: ik demonstreer eerst. Jullie kijken mee. Na de pauze gaan jullie zelf bouwen in je eigen project, en ik loop rond om vragen te beantwoorden.

[SLIDE 3]

Wat is Cursor eigenlijk? Het is een fork van Visual Studio Code. Dat betekent: alles wat je in VS Code kent — file explorer, terminal, debugger, extensies — werkt hier ook. Het belangrijke verschil: AI zit ingebouwd, niet als plug-in.

Dat is een groot verschil met bijvoorbeeld GitHub Copilot, waarbij je een extensie installeert. In Cursor is de AI overall: in de file-completion, in inline edits, in de chat, in code-review. En je kunt kiezen welk model je gebruikt — Claude, GPT, Gemini, of zelfs een lokaal model.

Waarom relevant voor jullie? Cursor is industry-standard aan het worden. Vacatures noemen het. En het is gratis Pro voor studenten via NOVI e-mail. Dat is de eerste reden waarom we hiermee starten.

BLOK 2 — Student Plan + Installatie

[SLIDE 4]

Het Student Plan. Wat krijg je: 500 fast requests per maand. De gratis tier krijgt er 50, dus dat is tien keer zoveel. Plus toegang tot premium modellen — Claude Opus, GPT-5, Gemini Pro. En tab-completion is altijd gratis, dus die telt niet mee.

Activeren: ga naar [cursor.com slash students](https://cursor.com/slash/students). Log in met je NOVI e-mailadres. Cursor stuurt je naar SheerID — dat is een externe verificatieservice — die bevestigt binnen één tot twee minuten dat je echt een NOVI-student bent. Daarna is je account upgraded.

(Ik laat de pagina cursor.com/slash/students zien.)

[SLIDE 5]

Installeren. Vier stappen. Download van cursor.com — dat herkent je OS automatisch. Installeer als gewone app — sleep naar Applications of next-next-finish op Windows. Open Cursor, kies "Sign in" bovenaan rechts. En optioneel: importeer je VS Code-instellingen. Cursor vraagt dit bij de eerste start. Klik "ja" — dan blijven al je keybindings, je extensies, en je theme behouden.

[SLIDE 6]

Eerste login en check of het Student Plan actief is. Settings — Cmd-comma. Tab "General". Daar staat "Account". Status moet zijn: "Pro Student".

(Ik laat dit zien in Cursor.)

Als het "Free" zegt — geen paniek. Log opnieuw in via [cursor.com slash students](https://cursor.com/slash/students). Soms moet je Cursor één keer herstarten zodat de upgrade in de IDE wordt doorgegeven. Als het na drie pogingen nog niet werkt: er is een handmatige route via support, maar die hebben we vandaag niet nodig.

BLOK 3 — Skills/Docs + Eerste project

[SLIDE 7]

Skills en Docs. Dit is een feature die Cursor onderscheidt van andere AI-editors. Je kunt Cursor framework-documentatie laten indexeren. Vervolgens kun je in chat zeggen "at-docs Next.js" en de AI weet exact wat de huidige Next.js docs zeggen — niet wat hij heeft geleerd in zijn training.

Setup: Settings, Features, Docs. Klik plus add new doc. Voeg toe: Next dot js, React, en Tailwind CSS. Cursor indexeert ze in één tot twee minuten.

Vanaf nu kun je dingen vragen als: "at-docs Tailwind hoe centreer ik een div?" en je krijgt het antwoord uit de officiële docs.

(Ik open Settings en demonstreer Docs.)

[SLIDE 8]

Eerste Next.js project aanmaken. Open een terminal in je web-folder. Type: `npx create-next-app` at latest mijn-eerste-cursor-app.

Beantwoord de prompts. TypeScript ja. ESLint ja. Tailwind ja. src dir ja. App Router ja. Import alias at-slash-asterisk — ja.

(Ik draai dit live in de terminal.)

Daarna `cd` in de folder, `npm run dev`, en je hebt localhost drie duizend draaiend. Dat is de Next.js startpagina. Standaard, maar werkend.

[SLIDE 9]

Open dit project in Cursor. File, Open Folder — kies mijn-eerste-cursor-app.

Wat je nu ziet: linker zijbalk is de file explorer, midden is de editor, onder is de terminal. Tot zover hetzelfde als VS Code.

Wat anders is: rechts is een AI-paneel. Je opent het met `Cmd-L`. Tab-completion is in Cursor slimmer dan in VS Code — hij kijkt mee met je hele bestand, niet alleen je laatste regel. En `Cmd-K` geeft je een AI-prompt direct in de editor.

Die drie verschillen zijn waar we het in de rest van de les over gaan hebben.

BLOK 4 — De drie sneltoetsen + request-management

[SLIDE 10]

De drie sneltoetsen die je vandaag leert. Tab voor auto-completion. `Cmd-K` voor inline edit. `Cmd-L` voor chat. Onthoud deze drie — dat is 80 procent van wat je nodig hebt om Cursor productief te gebruiken.

Tab is gratis — kost geen requests. `Cmd-K` en `Cmd-L` kosten elk één request, dus daar moet je iets meer mee opletten.

[SLIDE 11]

Tab. Het simpelste, maar onderschat. Cursor analyseert je hele bestand plus je recente edits. Hij suggereert hele blokken code, niet alleen één woord.

Een mooi voorbeeld: typ een commentaar zoals "slash slash render team list". Cursor begrijpt: aha, je wilt een lijst van team-leden. Hij genereert de hele component eronder. Druk Tab om te accepteren. Esc om te negeren.

Wanneer gebruik je Tab actief? Voor boilerplate — imports, types, props. Voor herhalende patronen — als je tien velden definieert in een form, voorspelt Cursor het patroon. Voor voorspelbare code — CRUD operations, lijsten renderen.

(Ik demonstreer Tab live: typ commentaar, krijg suggestie, druk Tab.)

[SLIDE 12]

`Cmd-K`. Inline edit. Dit is je gerichte AI-actie op een specifiek stuk code.

Selecteer code — of zet je cursor op een regel. Druk `Cmd-K`. Er opent een input-balkje boven je code. Typ wat je wilt — bijvoorbeeld "converteer naar TypeScript". Cursor toont een diff direct in de editor. Tab om te accepteren, Esc om te rejecten.

Wanneer `Cmd-K`? Eén variabele hernoemen. Eén functie refactoren. TypeScript-types toevoegen. Een comment toevoegen. Korte, gerichte wijzigingen waar je precies wilt zien wat er verandert.

Kosten: één request per `Cmd-K` aanroep.

(Ik demonstreer Cmd-K live op een kleine wijziging.)

[SLIDE 13]

Cmd-L. Chat. Voor grotere taken en vragen.

Cmd-L opent het chat-paneel rechts. Typ je vraag of taak. Je kunt context geven met at-symbolen: at-file foo punt tsx voor één bestand, at-docs Next dot js voor framework-docs, at-web voor live web search.

Cursor antwoordt en kan voorstellen om code te wijzigen. Je krijgt diffs te zien voordat ze worden toegepast.

Wanneer Cmd-L? Vragen — "hoe werkt useEffect?". Grote taken — "bouw een hele Hero-component". Code-review op een functie. Een soort senior collega in een chat-venster.

Kosten: één request per nieuwe chat-conversatie. Vervolg-berichten in dezelfde context zijn gratis tot je een nieuwe chat start.

(Ik demonstreer Cmd-L live met een vraag aan Cursor.)

[SLIDE 14]

Request management. Vijfhonderd requests per maand klinkt veel, maar als je niet oplet zijn ze in een week op.

Tips. Combineer taken in één Cmd-L. In plaats van vijf losse vragen, vraag in één prompt: "maak een Hero met titel, ondertitel, twee CTA-knoppen, en responsive design". Eén request, één goede output.

Gebruik Tab voor alles wat eenvoudig is — kost niets en is vaak even snel als typen.

Cmd-K alleen voor gerichte, specifieke één-bestand-edits. Niet voor "maak iets nieuws" — dat is Cmd-L territory.

Anti-pattern: tien losse Cmd-K's voor één feature. Dat zijn tien requests voor wat één Cmd-L had moeten zijn.

BLOK 5 — `.cursorrules`

[SLIDE 15]

Even kort over dot-cursorrules — kleine letter. Dit is een bestand in de root van je project. Cursor leest het bij elke AI-prompt. Het vertelt: "in dit project doen we het zo".

Bijvoorbeeld: "gebruik TypeScript strict mode". "Tailwind voor styling, geen CSS modules". "Server components by default, use-client alleen waar nodig". Imports met at-slash alias.

Het effect: je hoeft niet elke keer "vergeet niet TypeScript te gebruiken" in je chat te typen. Cursor weet het al.

Hoe maak je het? Vraag het Cursor zelf. Open een chat met Cmd-L: "Maak een dot-cursorrules voor dit Next dot js project". Kost één request, en je hebt een degelijke basis. Daarna pas je het naar smaak aan.

(Ik laat een voorbeeld zien.)

Belangrijk om te weten: in Les 14 — veel later — gaan we naar de nieuwere variant: een map dot-cursor-slash-rules met meerdere md-bestanden. Dat is hetzelfde idee maar uitgebreider. Voor nu is dot-cursorrules in de root prima.

BLOK 6 — Hands-on

[SLIDE 16]

Goed. Tot zover de theorie. Drie taken die jullie na de pauze gaan doen.

Taak één: bouw een Hero-component via Cmd-L. Titel, ondertitel, twee CTA-knoppen, gradient achtergrond, responsive.

Taak twee: tweak de styling via Cmd-K. Pak de twee knoppen, en verander kleur of spacing met een Cmd-K prompt.

Taak drie: kies één extra feature. Een FeatureCards-sectie, een ContactForm, of een Footer. Vrije keuze.

Daarna: git init, git add, git commit. Je eerste commit met Cursor-code.

Ik demonstreer nu taak één en taak twee, dan gaan we pauze houden, dan jullie aan de slag.

[SLIDE 17]

Taak één live. Ik open mijn Hero-component prompt en plak hem in Cmd-L.

(Ik plak en lees voor:)

"Bouw een Hero-component in app slash page punt tsx. Specs: gradient achtergrond in blauw, grote titel plus ondertitel, twee CTA-knoppen met de tekst 'Aan de slag' en 'Meer info', responsive met Tailwind, geen custom CSS."

Cursor genereert een diff. Ik bekijk hem — controleer of het responsive is, of de Tailwind klassen kloppen, of de structuur logisch is. Accept. Save. Naar browser, refresh.

(Ik refresh de browser en toon de Hero.)

Daar staat hij. Eén prompt, één goede Hero. Eén request gebruikt.

[SLIDE 18]

Taak twee live. Ik selecteer de twee knoppen in mijn code. Cmd-K. Input-balkje opent boven mijn selectie.

Ik typ: "verander primaire knop naar paars gradient, secundaire naar transparant met paarse rand".

Cursor toont een diff direct in de editor. Tab om te accepteren. Save. Naar browser, refresh.

(Ik toon het resultaat.)

Zelfde Hero, andere kleuren. Tweede request gebruikt.

[SLIDE 19]

Nu jullie. Open je eigen Next.js project in Cursor. Volg de stappen op de Lesopdracht-PDF. Drie taken: Hero, Cmd-K tweak, en een extra feature naar keuze.

Vijfenzeventig minuten. Vragen — gewoon stellen. Ik loop rond.

BLOK 7 — Afronding

[SLIDE 20]

Voor we afsluiten. Wat heb je vandaag geleerd?

Cursor is een VS Code-fork met AI ingebouwd. Je hebt Pro Student actief. Drie sneltoetsen — Tab gratis, Cmd-K één request, Cmd-L één request. En je hebt drie kleine componenten gebouwd met requests onder de tien.

Volgende les is Les 4 — TypeScript Fundamentals. Strict types, generics, utility types. We gaan ze toepassen in je Next.js project.

En veel later — Les 14 — duiken we diep in Cursor 3. De Editor Window versus de Agents Window. Agent mode en background agents. Vercel deployment met preview URLs.

Vandaag was de basis. Vanaf nu zie je deze sneltoetsen elke les terug. Tot volgende week.