



Les 5

Next.js Basics

Van React naar productie - we bouwen QuickPoll Part 1



Terugblik

Les 4 - TypeScript

Types

Primitives, arrays, interfaces

Unions + literal

string | number, 'pending' | 'done'

Narrowing

typeof, in, null checks, type predicates

Generics

Herbruikbare types met <T>

Vandaag: TypeScript in actie binnen Next.js.

Next.js = het React framework dat alle missende stukken toevoegt:

Routing, SSR, API routes, image optimization, deployment.

We bouwen samen QuickPoll - Part 1 vandaag, Part 2 in Les 6.

Planning

Vandaag - 180 minuten

Welkom + terugblik	10 min
Waarom Next.js + App Router	15 min
Server vs Client + data fetching	20 min
Routing + layouts + loading/error	15 min
Pauze	15 min
Klassikaal: QuickPoll Stap 0-3	95 min
Afsluiting + huiswerk	10 min

Theorie 1

Waarom Next.js?

React = UI library. Voor een echte app heb je meer nodig:

- Routing (URL -> pagina)
- Server-side rendering (SEO + snelle eerste paint)
- API endpoints
- Image optimization, fonts, caching
- Deployment

Next.js voegt dit toe bovenop React:

- v Industrie-standaard sinds 2022
- v Naadloos met Vercel deploy
- v Server Components - modern paradigm
- v TypeScript first-class
- v AI tools (Cursor, v0) zijn geoptimaliseerd voor Next.js

Theorie 2

App Router vs Pages Router

Pages Router (legacy)

pages/ folder
File = route
_app.tsx, _document.tsx
getServerSideProps

Niet meer leren als nieuw.

App Router (default)

app/ folder
File = route
Server Components default
Layouts, loading, error files
React Server Components

Wij gebruiken DEZE.

Project setup:

```
pnpm create next-app@latest quickpoll \
  --typescript --tailwind --app --no-src-dir
cd quickpoll && pnpm dev
```

Theorie 3

Folder-structuur + file-conventies

```
quickpoll/  
| - app/  
|   | - layout.tsx  
|   | - page.tsx      /  
|   | - globals.css  
|   | - about/page.tsx /about  
|   | - poll/[id]/page.tsx  
| - components/  
| - lib/  
| - public/  
| - next.config.ts
```

File-conventies:

page.tsx	De pagina-component
layout.tsx	Wraps children, persistent
loading.tsx	Loading state
error.tsx	Error boundary
not-found.tsx	Custom 404
route.ts	API endpoint

Routing rules:

app/about/page.tsx -> /about app/poll/[id]/page.tsx -> /poll/123 (dynamic)

(folder) = route group - geen URL-segment, alleen file-organisatie.

Theorie 4

Server vs Client Components

Server (default)

```
// app/page.tsx
export default async function Home() {
  const data = await fetch(...)
    .then(r => r.json());
  return <ul>{data.map(...)}</ul>;
}
```

Client ("use client")

```
"use client";
import { useState } from "react";

export function Counter() {
  const [n, setN] = useState(0);
  return <button>{n}</button>;
}
```

Wanneer welke:

Data fetching

Server

Client (kan, niet ideaal)

Hooks (useState)

x

ja

Event handlers

x

ja

Browser APIs (localStorage)

x

ja

Vuistregel: Server by default. "use client" alleen als nodig.

Theorie 5

Routing - dynamic + nested

```
// app/poll/[id]/page.tsx -> /poll/abc-123
```

```
export default async function Poll({  
  params,  
}: { params: Promise<{ id: string }> }) {  
  const { id } = await params;  
  return <div>Poll {id}</div>;  
}
```

Vanaf Next.js 16: params is een Promise (was object). Vergeet 'await' niet!

Link tussen pages:

```
import Link from "next/link";  
  
<Link href={` /poll/${id}`}>Bekijk poll</Link>
```

Theorie 6

Data fetching basics

In Server Component - gewoon await fetch():

```
async function getPolls() {  
  const res = await fetch("https://api.example.com/polls", {  
    next: { revalidate: 60 }, // cache 60 sec  
  });  
  return res.json();  
}
```

Cache opties:

```
fetch(url, { cache: "force-cache" }) // permanent (default)  
fetch(url, { next: { revalidate: 60 } }) // ISR  
fetch(url, { cache: "no-store" }) // nooit cachen
```

Voor live updates en interactiviteit: TanStack Query (komt later in Les 8).

Theorie 7

Loading + Error + 404 states

```
// app/poll/[id]/loading.tsx
export default function Loading() {
  return <div className="animate-pulse">Loading poll...</div>;
}
```

```
"use client";
// app/poll/[id]/error.tsx
export default function Error({ error, reset }: {
  error: Error; reset: () => void;
}) {
  return <button onClick={reset}>Probeer opnieuw</button>;
}
```

```
// app/poll/[id]/not-found.tsx
export default function NotFound() {
  return <h1>404 - Poll niet gevonden</h1>;
}
```

loading.tsx toont automatisch wanneer page-component nog `await` doet.



Pauze

15 minuten

Klassikaal

QuickPoll - wat bouwen we

QuickPoll = mini-app voor snelle peilingen. Next.js + TypeScript + Tailwind.

Part 1 (vandaag):

- v Setup project + folder structuur
- v Homepage met lijst polls (Server Component)
- v Hardcoded poll-data in lib/data.ts (in-memory)
- v Detail-pagina /poll/[id] met voting button
- v Voting werkt via button-click (lokale state)

Part 2 (Les 6):

- Server actions voor vote submission
- Loading + error states
- Polish UI + deployment naar Vercel

In Les 7 vervangen we lib/data.ts door echte Supabase queries.

Architectuur Part 1

```
app/  
|- page.tsx           # Homepage: lijst polls + create form  
|- poll/  
|  \- [id]/  
|     \- page.tsx      # Detail page met voting  
|- components/  
|  |- PollForm.tsx     # Client - create poll form  
|  \- VoteForm.tsx     # Client - stemmen  
|- lib/  
|  \- data.ts          # In-memory poll store
```

Aanpak vandaag:

- Tim codet op het scherm
- Studenten volgen mee in eigen project
- Cursor Tab-completion mag
- Vraag-stop momenten elke 15-20 min

Als je verdwaalt: roep Tim/buurman aan. Iedereen moet meekomen.

Snelle ref

Tailwind cheatsheet

```
<div className="flex items-center gap-4 p-4 bg-white rounded-lg shadow">  
  <h2 className="text-2xl font-bold text-gray-900">Hello</h2>  
  <p className="text-sm text-gray-500">World</p>  
</div>
```

Responsive (mobile first):

```
<div className="text-sm md:text-base lg:text-lg">
```

Hover, focus, dark:

```
<button className="bg-blue-500 hover:bg-blue-600 focus:ring-2 dark:bg-blue-700">
```

99% van de tijd geen custom CSS nodig. Tailwind doet het allemaal.

Voor Les 6 (QuickPoll Part 2)

1. QuickPoll Part 1 afmaken

Stap 0-3 thuis afronden als je vastliep

2. Voeg een /about pagina toe

Gebruik route group (marketing) - oefen layouts

3. 2 polls hardcoded in lib/data.ts

Minstens 3 opties per poll

4. Inleveren via Teams

GitHub repo + screenshot van werkende app

Tip: Cursor + AGENTS.md = goede 'use client' / Server Components conventies vastleggen.

Volgende les

Les 6 - QuickPoll Part 2

We maken QuickPoll af:

- Server Actions voor vote submission
- Loading / error / not-found states
- Polish UI met Shadcn/ui
- Deploy naar Vercel (live URL!)

Vorbereiden:

QuickPoll Part 1 draait op localhost - Stap 0-3 af.

Vragen?

Tot volgende week!