

Les 6 — Lesstof

Next.js QuickPoll Compleet

Vak: Technical Foundations
Cursus: AI Developer
Volgende les: Les 7 — Supabase Setup

Vak: Technical Foundations **Vorige les:** Les 5 — Next.js Basics **Volgende les:** Les 7 — Supabase Setup

Inhoud

1. [Waar staan we?](#1-waar-staan-we)
2. [Server Actions — forms zonder API routes](#2-server-actions--forms-zonder-api-routes)
3. [Forms in Next.js](#3-forms-in-nextjs)
4. [revalidatePath — UI bijwerken](#4-revalidatepath--ui-bijwerken)
5. [Loading + Error states uitwerken](#5-loading--error-states-uitwerken)
6. [Middleware basics](#6-middleware-basics)
7. [Environment variables](#7-environment-variables)
8. [API routes — wanneer ze handig zijn](#8-api-routes--wanneer-ze-handig-zijn)
9. [Polish — UX details die ertoe doen](#9-polish--ux-details-die-ertoe-doen)
10. [Voorbereiden op Supabase (Les 7)](#10-voorbereiden-op-supabase-les-7)

1. Waar staan we?

In Les 5 bouwden we QuickPoll Part 1:

- Homepage met lijst polls + create form
- Detail-pagina met voting
- Alles in-memory (`lib/data.ts`)

Wat nog mist:

- Stemmen werkt niet écht — refresh = data weg
- Form-handling is matig

- Geen loading states
- Geen error handling
- Data verdwijnt bij server-restart

Deze les: alles aanscherpen + voorbereiden voor Supabase in Les 7.

2. Server Actions — forms zonder API routes

Next.js 14+ introduceerde **Server Actions** — async functies die je rechtstreeks vanuit een form aan de server kunt aanroepen. Geen API route nodig.

Basis

```
// app/page.tsx
async function createPoll(formData: FormData) {
  "use server";

  const title = formData.get("title") as string;
  // ... insert in database
}

export default function Page() {
  return (
    <form action={createPoll}>
      <input name="title" required />
      <button>Create</button>
    </form>
  );
}
```

"use server" markeert deze functie als server-side. Form posts er rechtstreeks naar.

Voordelen

- Minder boilerplate (geen API route)
- Progressieve enhancement (werkt zonder JS)
- Type-safe — params zijn FormData
- Te combineren met `revalidatePath` voor UI updates

3. Forms in Next.js

useFormStatus + useFormState

Voor loading + error states in client components:

```
"use client";
import { useFormStatus } from "react-dom";

function SubmitButton() {
  const { pending } = useFormStatus();
  return <button disabled={pending}>{pending ? "Bezig..." : "Vote"}</button>;
}

<form action={voteAction}>
  <input name="option" />
  <SubmitButton />
</form>
```

Validation

Zod is de standaard:

```
import { z } from "zod";

const schema = z.object({
  title: z.string().min(3).max(100),
  options: z.array(z.string().min(2)),
});

async function createPoll(formData: FormData) {
  "use server";
  const parsed = schema.safeParse({
    title: formData.get("title"),
    options: formData.getAll("options"),
  });
  if (!parsed.success) return { error: parsed.error.message };
  // ... save
}
```

4. revalidatePath — UI bijwerken

Na een mutation moet je gecachte data invalideren — anders ziet user oude versie.

```
import { revalidatePath } from "next/cache";

async function vote(formData: FormData) {
  "use server";
  // ... save vote
  revalidatePath(`/poll/${id}`); // refresh poll page
  revalidatePath("/");           // refresh homepage
}
```

Alternatieven

- `revalidateTag("polls")` — invalideer alle fetches met tag "polls"
- `redirect("/somewhere")` — redirect after action (impliciet revalidate)

5. Loading + Error states uitwerken

Skeleton loaders

In plaats van "Loading..." → skeleton matching de UI:

```
// app/poll/[id]/loading.tsx
export default function Loading() {
  return (
    <div className="space-y-4 p-6">
      <div className="h-8 bg-gray-200 rounded animate-pulse w-1/2" />
      <div className="h-32 bg-gray-200 rounded animate-pulse" />
    </div>
  );
}
```

Voelt sneller dan een spinner.

Error boundary met retry

```
"use client";
export default function Error({ error, reset }: {
  error: Error & { digest?: string };
  reset: () => void;
}) {
  return (
    <div className="p-6 bg-red-50 rounded">
      <h2 className="text-red-900 font-bold">Iets ging mis</h2>
      <p className="text-sm">{error.message}</p>
      <button onClick={reset} className="mt-4 text-blue-600 underline">
        Probeer opnieuw
      </button>
    </div>
  );
}
```

6. Middleware basics

`middleware.ts` in project-root = runt voor elke request. Voor auth, redirects, headers.

Voorbeeld — protect /admin

```
// middleware.ts
import { NextResponse, type NextRequest } from "next/server";

export function middleware(request: NextRequest) {
  const isAdminPath = request.nextUrl.pathname.startsWith("/admin");
  const isAuthenticated = request.cookies.has("session");

  if (isAdminPath && !isAuthenticated) {
    return NextResponse.redirect(new URL("/login", request.url));
  }
}

export const config = {
  matcher: "/admin/:path*",
};
```

In Les 9 gebruiken we dit voor Supabase Auth.

7. Environment variables

`.env.local`

```
DATABASE_URL=postgres://...
NEXT_PUBLIC_APP_URL=http://localhost:3000
```

Server-only vs client-available

Prefix	Beschikbaar	Voorbeeld
(geen prefix)	Alleen server	DATABASE_URL
NEXT_PUBLIC_	Server + client	NEXT_PUBLIC_API_URL

Belangrijke regel: secrets (API keys, DB passwords) hebben **NOOIT** NEXT_PUBLIC_ prefix. Anders staat het in je client JS-bundle.

Vercel

In Vercel dashboard: Settings → Environment Variables. Apart voor Production / Preview / Development.

8. API routes — wanneer ze handig zijn

App Router heeft ook API routes (`route.ts`). Wanneer wel/niet?

Server actions hebben de voorkeur

Voor forms binnen je app → server actions. Minder boilerplate.

API routes voor

- Webhooks (Stripe events, GitHub events)
- Third-party services die HTTP endpoints verwachten
- Mobile app calls
- Cron jobs

Voorbeeld

```
// app/api/health/route.ts
export async function GET() {
  return Response.json({ status: "ok", time: new Date() });
}

// app/api/polls/route.ts
export async function POST(request: Request) {
  const data = await request.json();
  // ... validate, save
  return Response.json({ ok: true });
}
```

9. Polish — UX details die ertoe doen

Feedback bij submit

Geen feedback = user klikt twee keer. Disable button + spinner + success-toast.

```
const { pending } = useFormStatus();
<button disabled={pending}>
  {pending ? "Bezig..." : "Stemmen"}
</button>
```

Optimistic updates

User klikt → UI update direct → fetch op achtergrond. Voelt instant.

```
import { useOptimistic } from "react";

const [optimisticVotes, addOptimisticVote] = useOptimistic(
  votes,
  (current, newVote) => [...current, newVote]
);
```

Empty states

Geen polls? Toon een mooie empty state + CTA: "Maak je eerste poll".

Toasts

react-hot-toast of sonner (komt met shadcn/ui). Goedkope feedback.

10. Voorbereiden op Supabase (Les 7)

Onze QuickPoll heeft één probleem: data is in-memory. Restart = weg.

Wat verandert in Les 7

- `lib/data.ts` wordt `lib/supabase.ts` (echte client)
- In-memory arrays → database tabellen
- Server actions roepen `supabase.from("polls").insert(...)` aan

Architectuur na Les 7

```
app/  
  page.tsx           # Server Component – fetch polls van Supabase  
  poll/  
    [id]/  
    page.tsx         # Server Component – fetch by id  
  lib/  
    supabase.ts      # Supabase client
```

De rest van je code (page-structuur, components, server actions) blijft hetzelfde.

Bronnen

- **Next.js Server Actions:** <https://nextjs.org/docs/app/getting-started/updating-data>
- **Next.js Forms guide:** <https://nextjs.org/docs/app/guides/forms>
- **Zod:** <https://zod.dev>
- **Next.js caching:** <https://nextjs.org/docs/app/guides/caching>
- **Loading UI:** <https://nextjs.org/docs/app/api-reference/file-conventions/loading>
- **Error Handling:** <https://nextjs.org/docs/app/getting-started/error-handling>