Next JS Notes - Day 3

1️⃣ Setting up a SQLite Database (just understand the basics no need to go deeper for now)

We'll use **better-sqlite3** (fast + simple) for a local database.

Install

```
npm i better-sqlite3
```

Creating a Table

```
db.prepare(`
  CREATE TABLE IF NOT EXISTS blogs (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    title TEXT NOT NULL
  )
`).run();
```

- `CREATE TABLE IF NOT EXISTS` → only creates if not already present.
- Columns:
  - `id` → Auto-increment primary key.
  - `title` → Text column (required).

Inserting Data

```
function initData(data: string[]) {
  const stmt = db.prepare(`
    INSERT INTO blogs (title) VALUES (?)
  `);

  for (const val of data) {
    stmt.run(val);
  }
}
```

Fetching Data

```
// Insert
db.prepare("INSERT INTO blogs (title) VALUES (?)").run("My first blog");

// Fetch all rows
const all = db.prepare("SELECT * FROM blogs").all();

// Fetch single row
const one = db.prepare("SELECT * FROM blogs WHERE id = ?").get(1);
```

2️⃣ Async Server Components for Data Fetching

In the **App Router**, all components are **Server Components by default**, and they can be **async**.

This makes it super easy to fetch data directly inside the component.

👉 Example:

```
// app/blogs/page.tsx
import db from "@/lib/db";

export default async function BlogsPage() {
  const blogs = await db.prepare("SELECT * FROM blogs").all();
  return (
    <div>
      <h1>All Blogs</h1>
      <ul>
        {blogs.map((b: any) => (
          <li key={b.id}>{b.title}</li>
        ))}
      </ul>
    </div>
  );
}
```

- `async` is allowed because server components can run **on the server** before rendering.
- ✅ No need for `useEffect` or `fetch` in client — the data is already ready in the HTML.

3️⃣ Special Files in Next.js (App Router)

1. `loading.tsx`

- Renders when a page (or layout) is **loading**.
- Useful for **skeletons, spinners, shimmer effects**.

```
// app/blogs/loading.tsx
export default function Loading() {
  return <p>⏳ Loading blogs...</p>;
}
```

---

2. `error.tsx`

- Renders if a page (or layout) throws an **error**.
- Must be a **Client Component**.

```
"use client";

export default function Error({ error, reset }: { error: Error; reset: () ⇒ void }) {
  return (
    <div>
      <h2>❌ Something went wrong!</h2>
      <p>{error.message}</p>
      <button onClick={() ⇒ reset()}>Try again</button>
    </div>
  );
}
```

3. `not-found.tsx`

- Renders if a page is not found (`404`).
- Can be used in the root `app/` or inside a nested route.

```
// app/not-found.tsx
export default function NotFound() {
  return <h2>⚠️ Page not found</h2>;
}
```

4️⃣ Summary

- **SQLite** → quick, local database for testing.
- **Async Server Components** → fetch DB data directly in the component.
- **loading.tsx** → show while page is loading.

- **error.tsx** → handle runtime errors.
- **not-found.tsx** → custom 404 pages.