What is NextJS?

And why would you use it?

A React Framework

For Building **Fullstack** React Apps

Maximilian Schwarzmüller — “NextJS - The Complete Guide”

Why do we need a
React framework?

**Isn’t React itself
already a library?**

NextJS is a fullstack React framework

It vastly **simplifies** the process of building fullstack applications with React

NEXT.JS

Trend: Build fullstack apps instead of SPAs

Get the **best of both worlds**:

Highly interactive frontends,
blending seamlessly with
tightly connected backends

React is becoming a hybrid library

Improved server-side rendering

Streaming responses

React Server Components

Server Actions

Using these features without a framework is tricky & typically not all you need

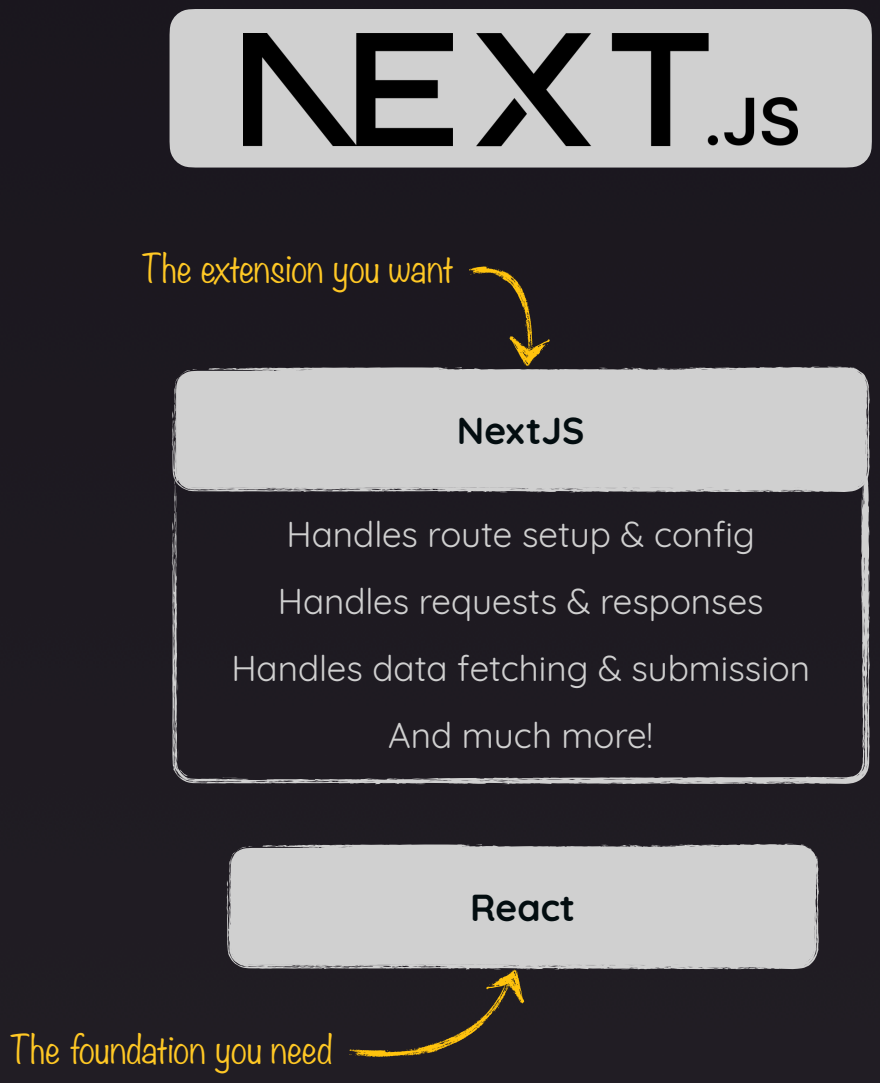
Route Setup & Handling

Form Submission

Data Fetching

Authentication

And much more!


NEXT.JS
The extension you want
NextJS
Handles route setup & config
Handles requests & responses
Handles data fetching & submission
And much more!
React
The foundation you need

Key Features & Benefits

Fullstack Apps

NextJS blends frontend + backend (in the same project)

Advantage: Frontend and backend tasks are part of the same project

File-based Routing

Routes are configured via the filesystem (folders + files)

Advantage: No code-based configuration or extra packages for routing required

Server-side Rendering

By default, NextJS (pre-) renders all pages on the server

Advantage: The finished HTML page (incl. content) is sent to the client (→ great for SEO)

Two Approaches For Building NextJS Apps

Pages Router

Has been around for many years

Very stable

Used in many existing NextJS projects

Allows you to build feature-rich fullstack apps with React

App Router

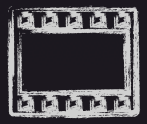
Introduced with NextJS 13

Marked as stable but still relatively new & partially buggy

Supports modern Next & React features (→ fullstack React apps)

The future of NextJS

How To Get The Most Out Of This Course



Watch The Videos

Watch them at your pace!
(use the video player controls)

Pause & rewind if needed

Take your time!

Practice!

Pause videos & try the next steps on your own

Build demo projects & revisit course projects

Support

Use the code snapshots & attachments

Ask & answer in the Q&A section

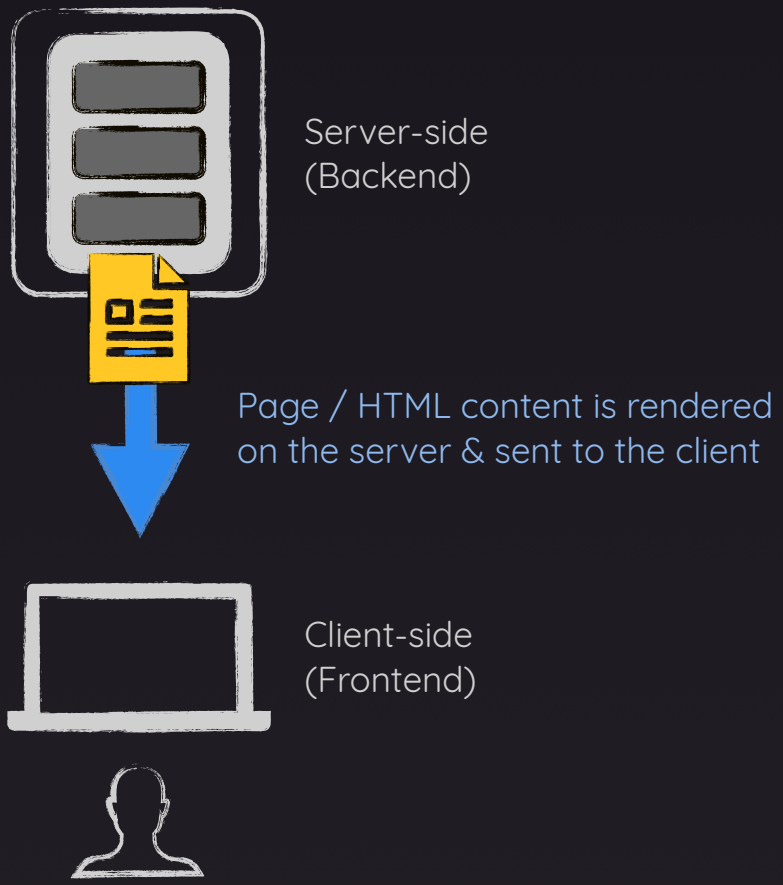
Find like-minded developers on Discord

NextJS Essentials

The Core Concepts You Must Know

- Routing, Pages & Components
- Fetching & Sending Data
- Styling, Images & Metadata

NextJS Renders Pages On The Server



Vanilla React Apps Render On The Client

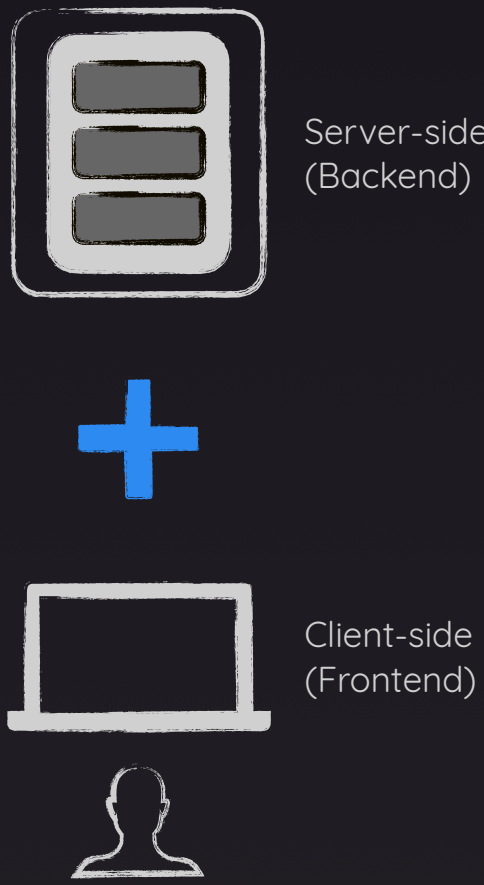
Server-side
(Backend)

Only returns one single HTML file which contains the client-side JS code

Client-side
(Frontend)

The visible content is generated & rendered on the client-side (by the client-side React code)

With NextJS, You Build Fullstack Applications



Filesystem-based Routing

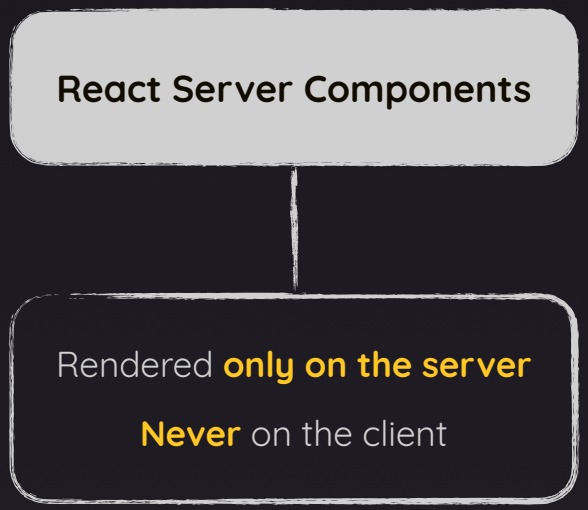
NextJS uses files & folders to define routes

Only files & folders inside the “app” folder are considered!

NextJS Works With React Server Components

Components which require a special “environment”

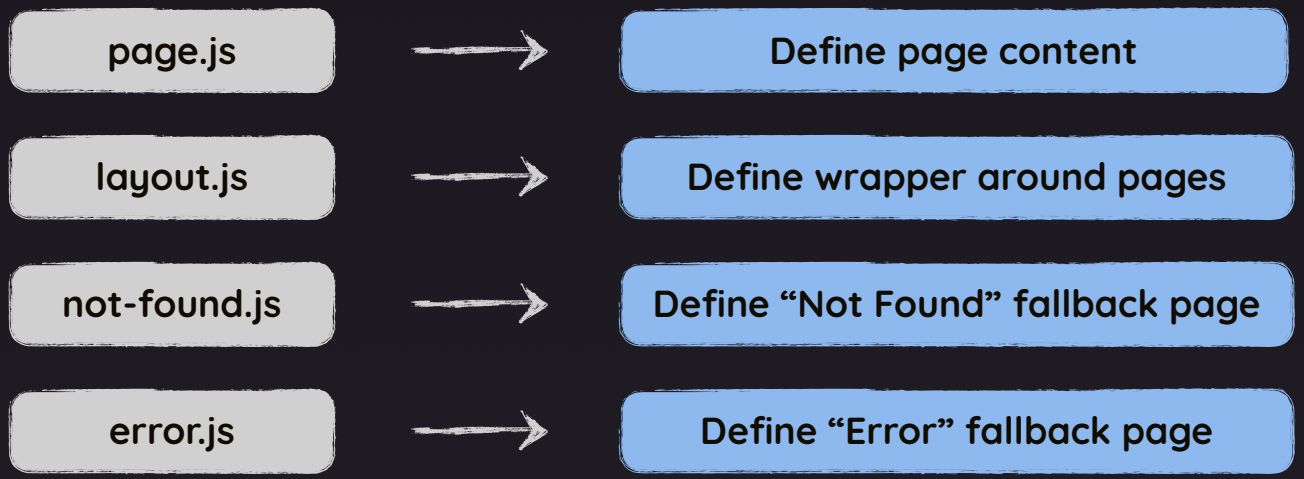
NextJS provides such an environment



Filenames Matter!

NextJS relies on reserved, special filenames

But the filenames only matter inside the “app” folder

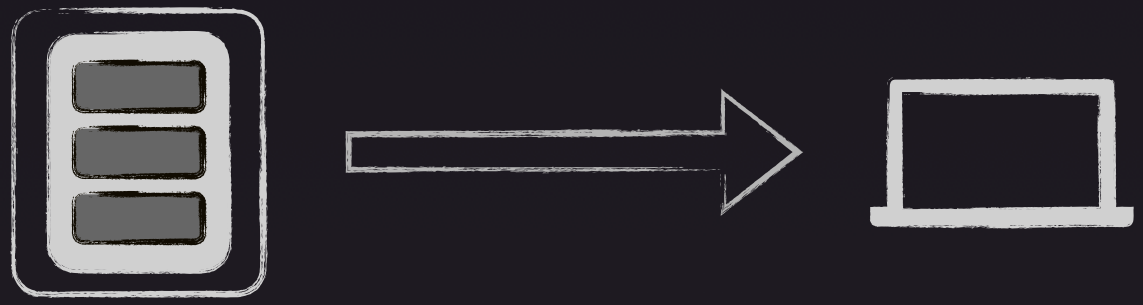


And others — covered later!

Server- & Client-side Working Together

Server-side
(Backend)

Client-side
(Frontend)



The backend **executes the server component functions** & hence derives the to-be-rendered HTML code

The client-side **receives & renders** the to-be-rendered HTML code

Exercise

Create three routes

/meals

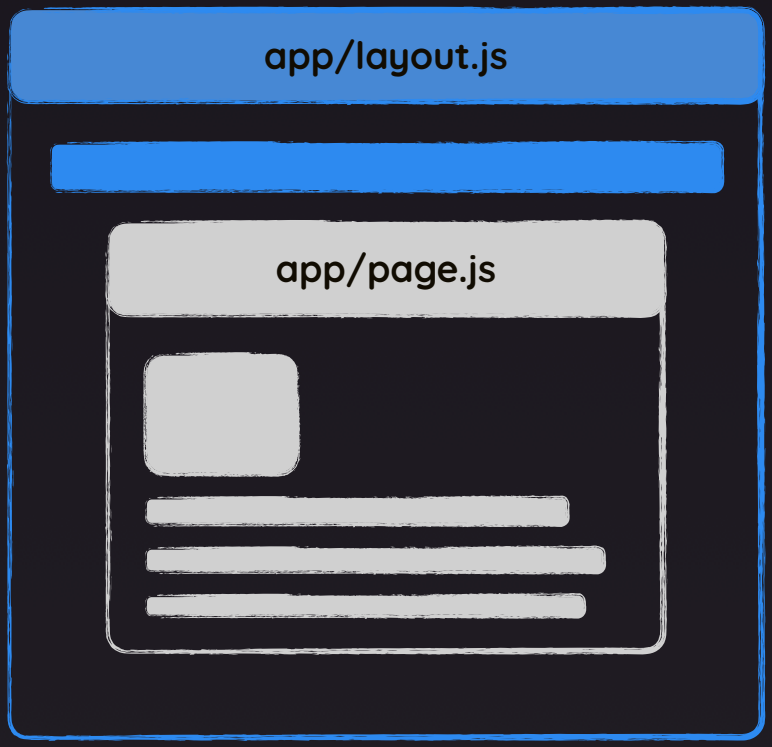
/meals/share

/community

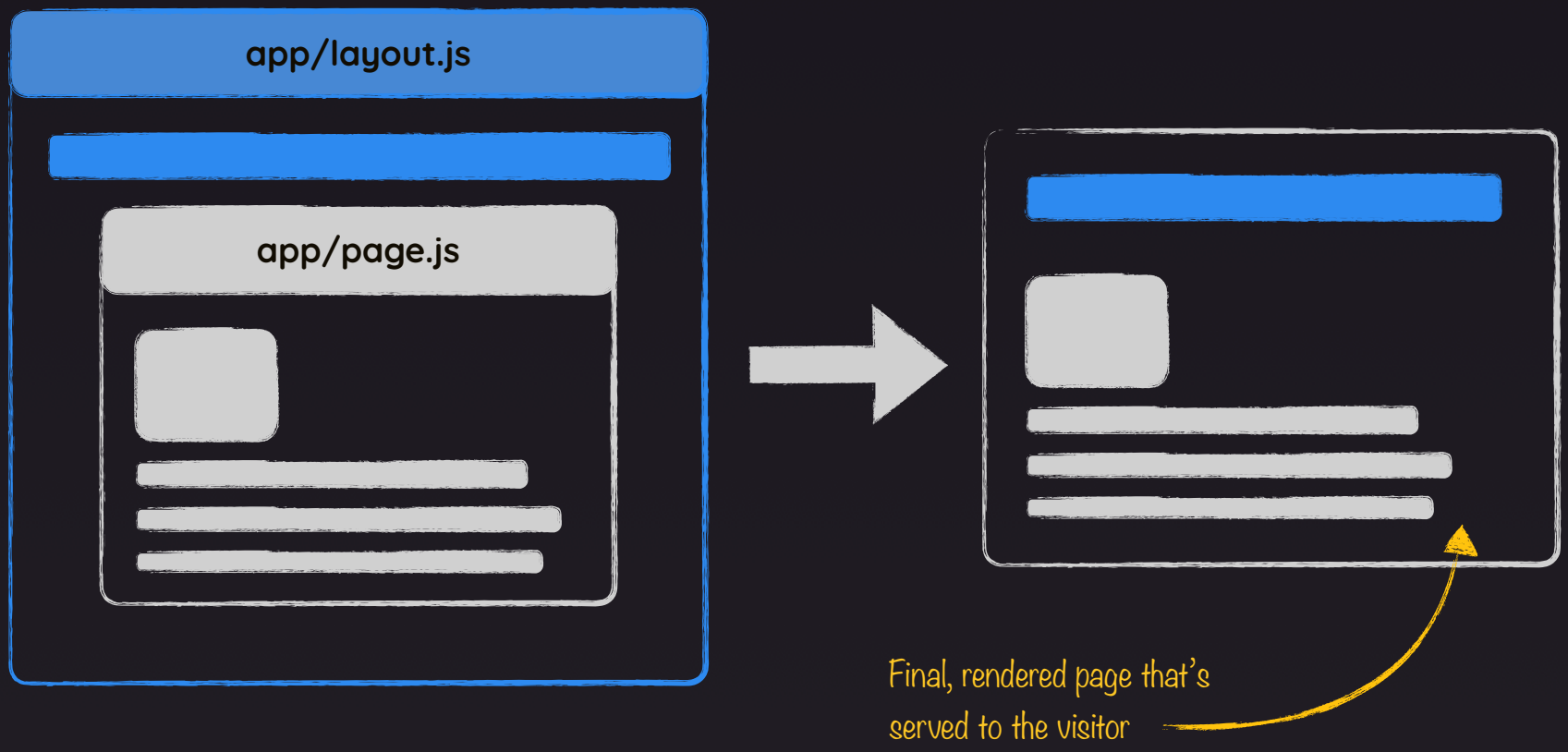
Create a dynamic route

/meals/<some slug>

Pages & Layouts



Pages & Layouts



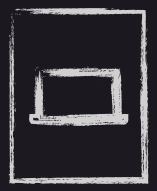
Server vs Client Components

React Server Components (RSC)

Components that are **only** rendered on the server

By default, all React components (in NextJS apps) are RSCs

Advantage: Less client-side JS, great for SEO



Client Components

Components that are **pre-rendered** on the server but then also **potentially on the client**

Opt-in via “use client” directive

Advantage: Client-side interactivity