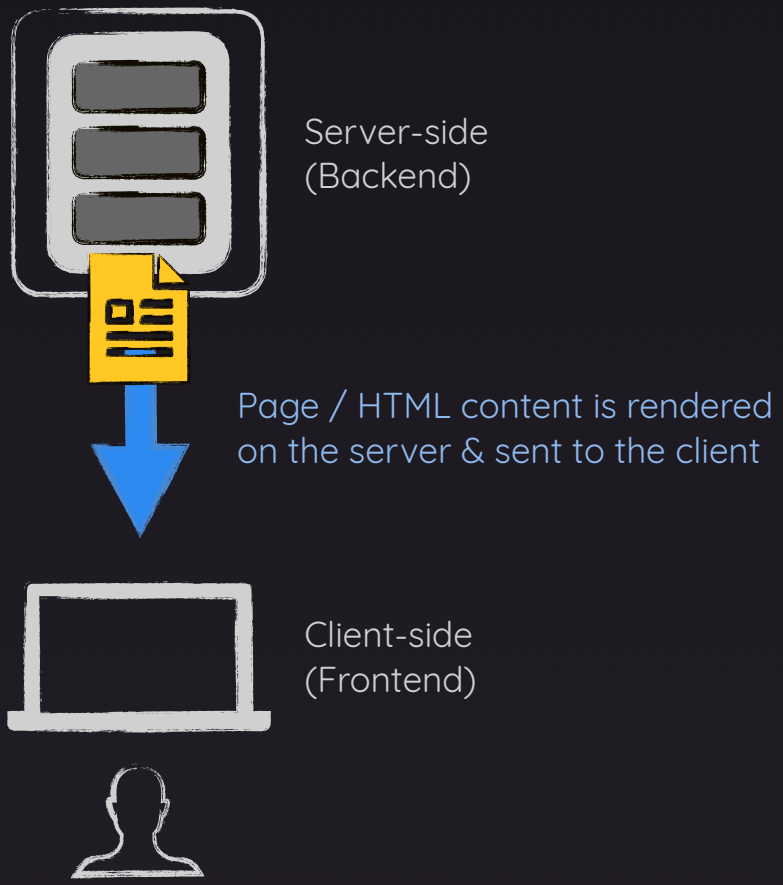
NextJS Essentials

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)

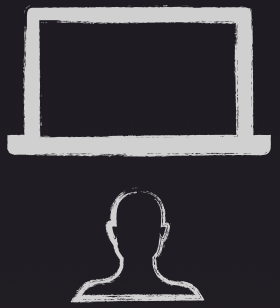
Only returns one single HTML file which contains the client-side JS code

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

Server-side
(Backend)



Client-side
(Frontend)

Filesystem-based Routing

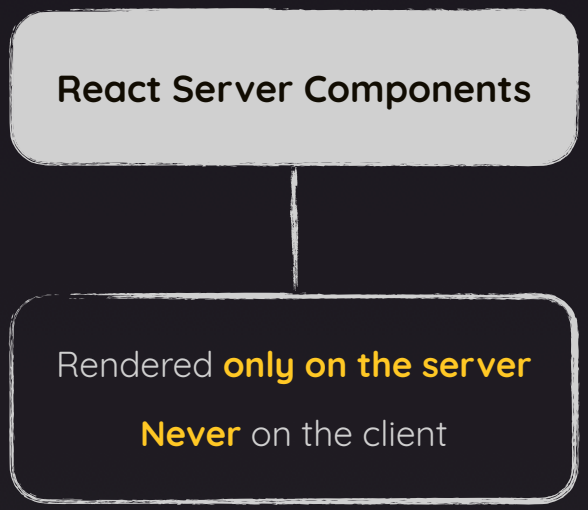
NextJS uses files & folders to define routes

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

NextJS Works With React Server Components

Components which require a special “environment”

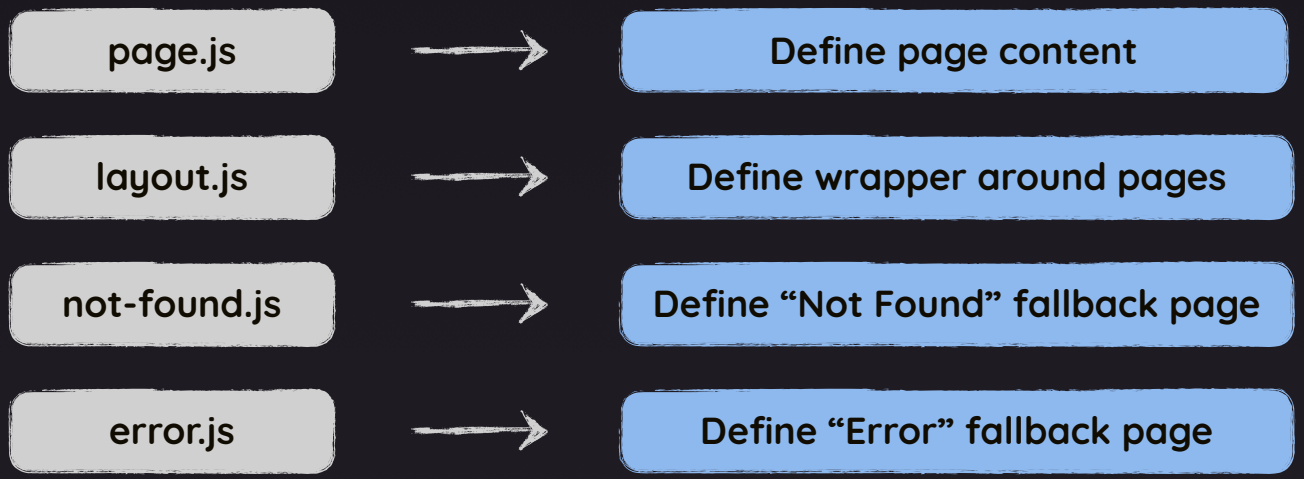
NextJS provides such an environment



Filenames Matter!

NextJS relies on reserved, special filenames

But the filenames only matter inside the “app” folder

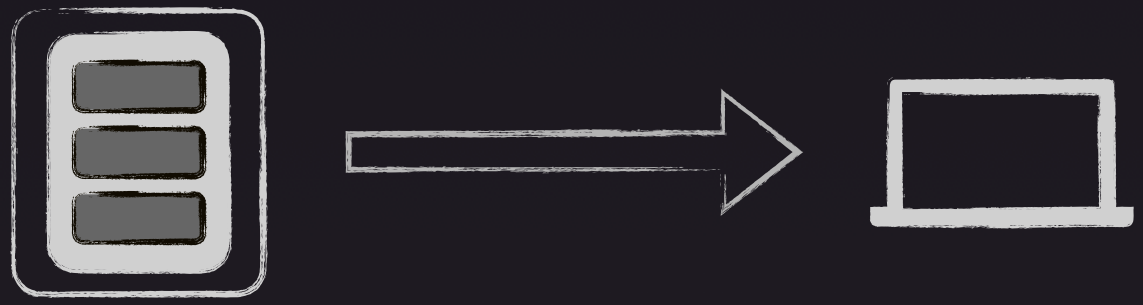


And others — covered later!

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

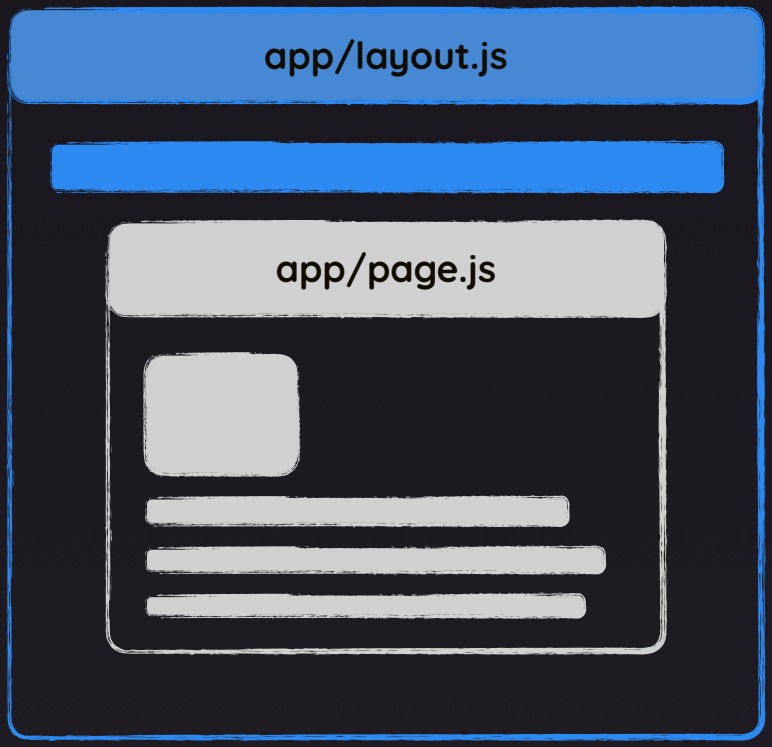
/meals/share

/community

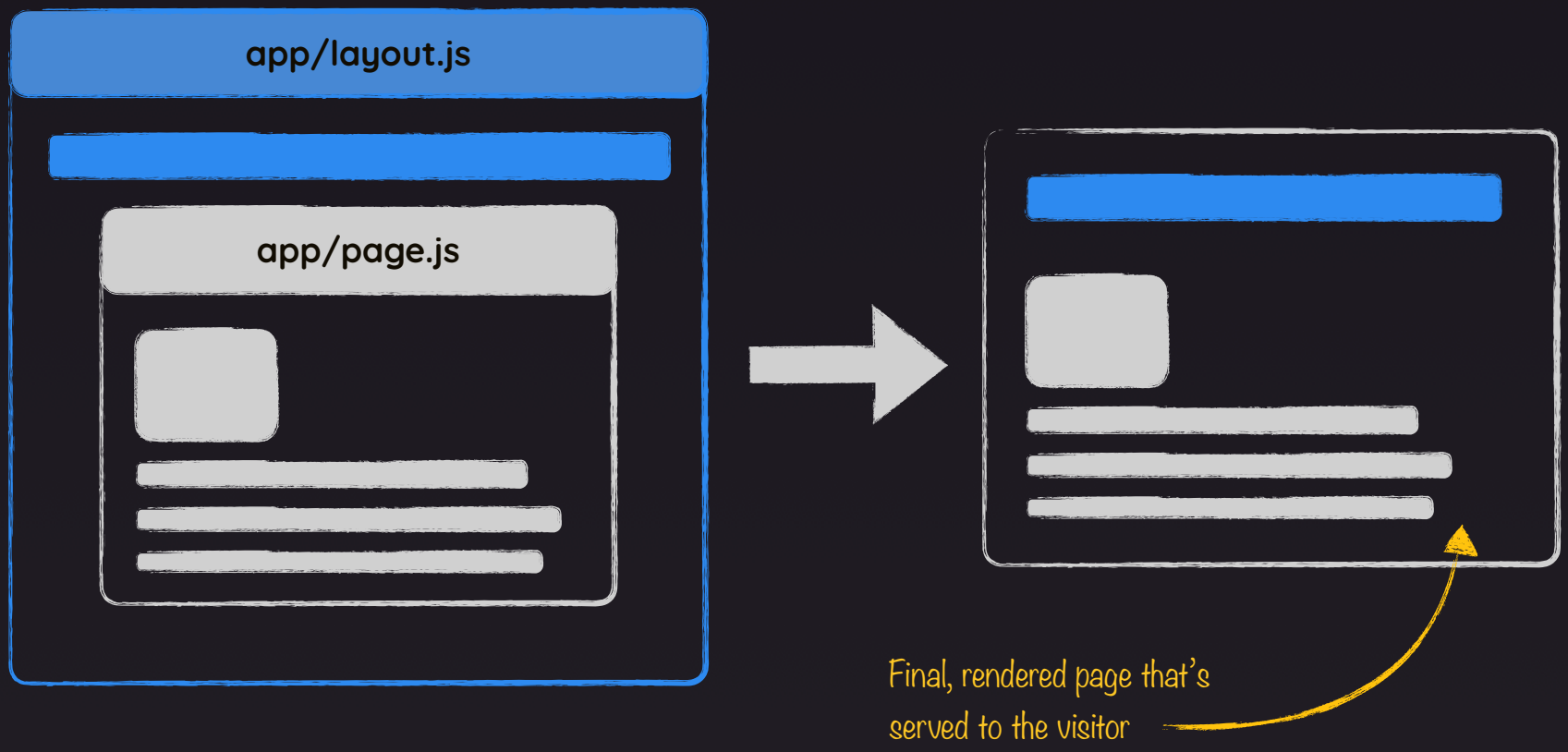
Create a dynamic route

/meals/<some slug>

Pages & Layouts



Pages & Layouts



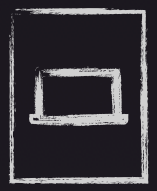
Server vs Client Components

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

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

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)

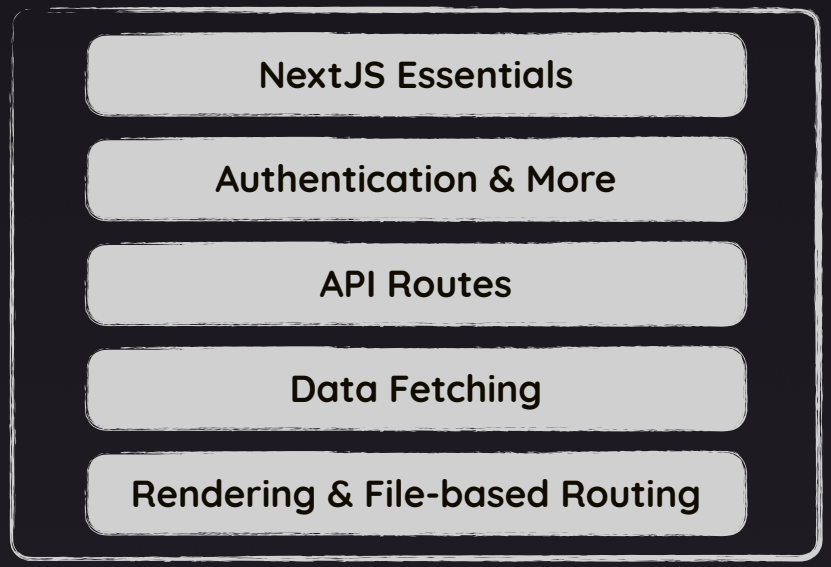
Advantage: No code-based configuration or extra packages for routing required

Server-side Rendering

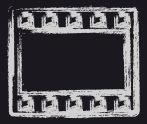
By default, NextJS (pre-) renders all pages on the server

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

About The Course



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

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

App Router NextJS Essentials

Routing & Page Rendering

React Server Components vs Client Components

Styling

Working with Images & Metadata

Data Fetching

Data Mutation & Server Actions

File Upload

Maximilian Schwarzmüller — “NextJS - The Complete Guide”

App Router → Pages Router

I added the App Router section after the initial course release

Learning about the Pages Router can still make a lot of sense!

Routing & Page Rendering

File-based Routing, Component Types & More

- Understanding **Routing** In NextJS Applications
- **File Name** Conventions & Project **Structure**
- **Server Components** vs **Client Components**

A Challenge For You

Add two new pages

/news → **Show a list of news item links**

(for this exercise, it's enough to just output some dummy text)

/news/<id> → **Show a detail page for a news item**

(for this exercise, it's enough to just output some dummy text)

Also add a <MainHeader> component that contains links to the (already existing) "Home" page and the newly added "News" page

Standard Routes

/about

Folder name = route path

app/about ⟶ my-website.com/about

Dynamic Routes

/posts/<dynamic>

Define dynamic segments by wrapping the folder name with []

app/posts/[slug] ⟶ my-website.com/posts/next-is-awesome

Layouts & Pages

Pages

Created via **page.js** file

Define page content (JSX) for a route

Layouts

Created via **layout.js** file

Define wrapping layout for one or more pages

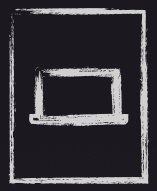
Server vs Client Components

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

Not Found & Errors

Not Found Pages

Created via **not-found.js** file

Shown if a “Not Found” (404) error occurred

Error Fallback Pages

Created via **error.js** file (or **global-error.js**)

Shown if an error gets thrown by a child page or layout

Loading UI

Loading Fallback Page

Created via **loading.js** file

Shown if the child page is still loading data

Route Handlers (API Routes)

Non-Page Routes

Created via **route.js** file

Manually handle request & send
(any) response

Caching

Non-Page Routes

Created via **route.js** file

Manually handle request & send
(any) response

Parallel Routes

/dashboard

Load multiple pages in one page by prefixing folder names with @

app/dashboard/@analytics
app/dashboard/@expenses

my-website.com/dashboard

Intercepting Routes

/dashboard

Intercept navigation requests by prefixing folder names with (.), (..) etc

app/expenses/(.)new ⟶ my-website.com/expenses

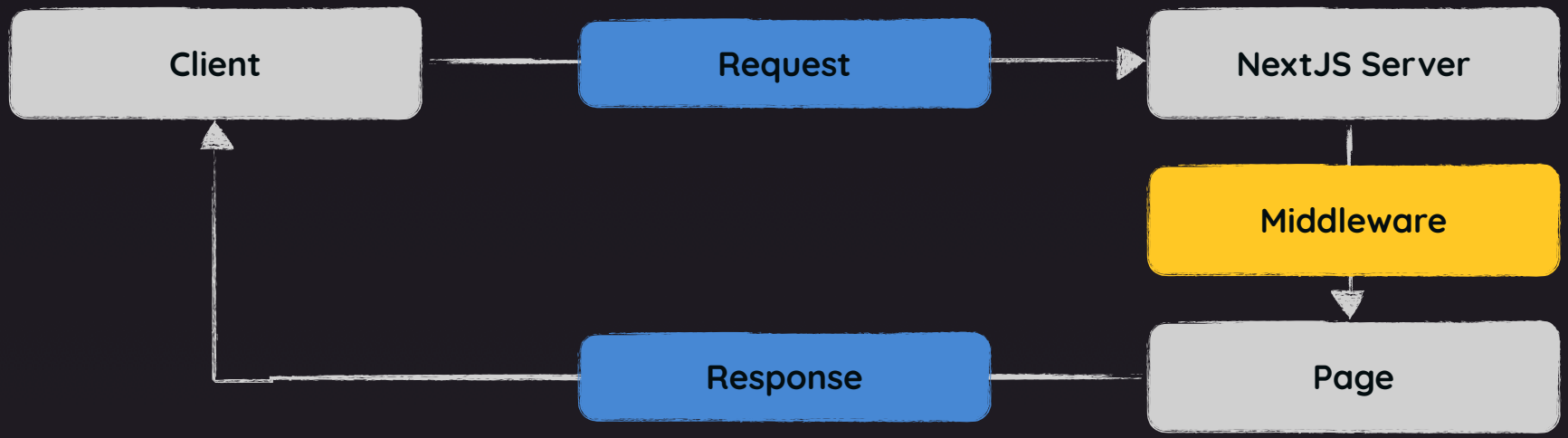
Route Groups

/posts vs /welcome

Define groups by wrapping folders with ()

app/(marketing)/welcome ⟶ my-website.com/welcome

Middleware



Data Fetching

Fetching & Using Data From Different Sources

- **Where & How** To Fetch Data
- Working with **APIs**
- Loading Data from **Databases & Other Sources**

Data Mutation

Sending, Storing & Changing Data

- Data Mutation **Options**
- Working with **Server Actions**

There Are Multiple Data Mutation Strategies

Option 1

Standalone Backend

A separate backend receives requests from the NextJS app and stores / mutates all data

Problem: Unnecessary complexity, unnecessary extra server

Option 2

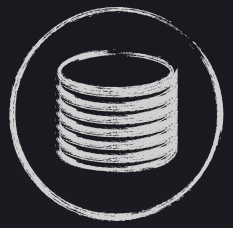
Integrated NextJS App

The NextJS app itself contains the data mutation code and reaches out directly to the data stores

NextJS App

Renders pages & handles form submissions / data mutation

Advantage: Reduced complexity, single codebase & server



Caching

How NextJS Optimizes Data Fetching & Page Rendering

- Understanding NextJS Caching
- Revalidating Cached Data
- Manually Caching Data

NextJS Performs Aggressive Caching

Request Memoization

NextJS stores data requests with the same configuration

This avoids unnecessary duplicate data fetches

Cache only persists during request duration

Data Cache

NextJS stores & reuses fetched data until it's revalidated

This avoids unnecessary requests to the data source & speeds up the application

The cache persists until it's revalidated (manually or after a set time)

Full Route Cache

NextJS stores the rendered HTML & RSC at build time

This avoids unnecessary HTML render cycles & data fetches

The cache persists until the related data cache is revalidated

Router Cache

NextJS stores the RSC payload in memory in the browser

This ensures extremely fast page transitions since no server request is needed

Maximilian Schwarzmüller — "NextJS - The Complete Guide"

Optimizations

Working on SEO & Performance

- Image Optimizations
- Adding Page Metadata

Authentication

Logging Users In & Out

- User Signup
- User Login
- Protecting Routes

How Does Authentication Work?

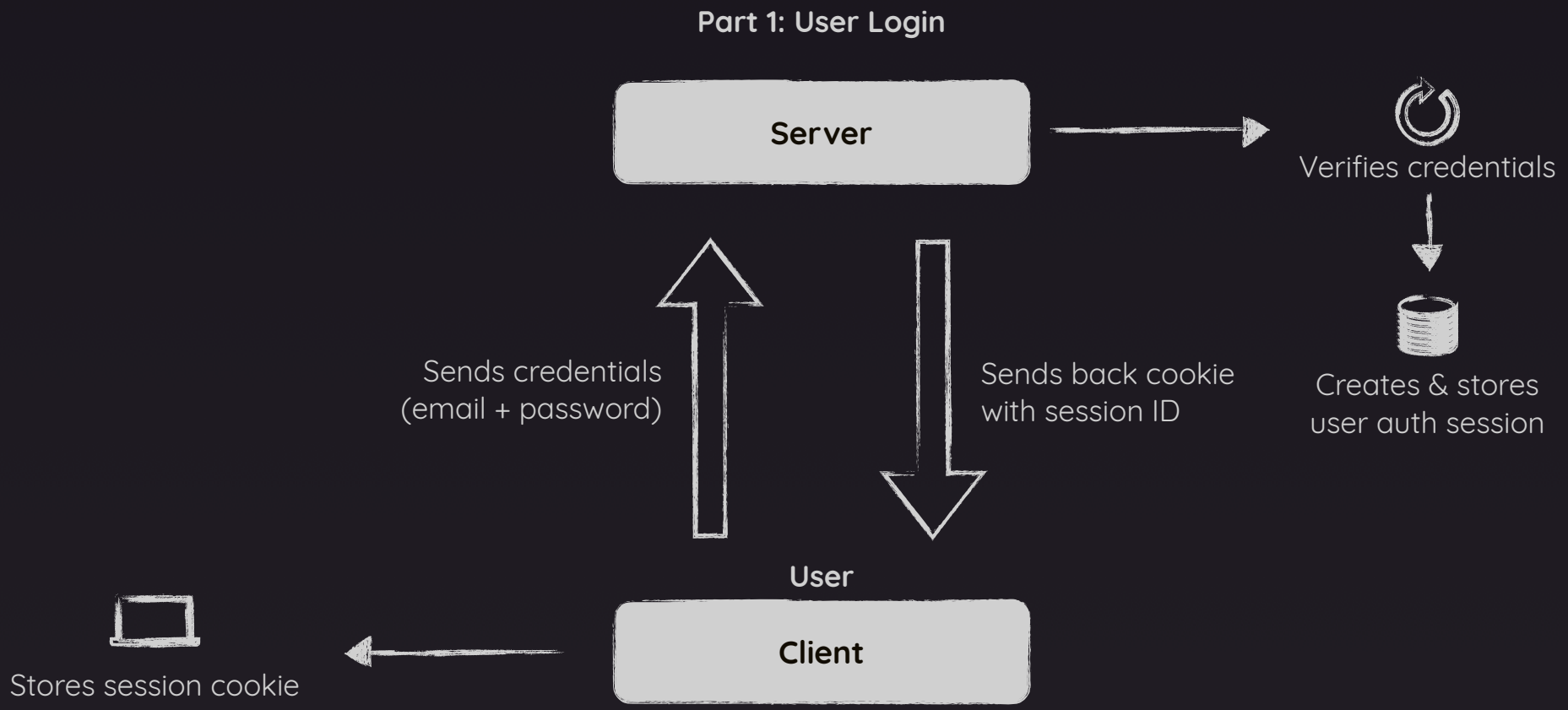
Login

Verify user credentials & “mark” user as authenticated

Authorized Access

Grant authenticated user access to protected routes & resources

How Does Authentication Work?



How Does Authentication Work?

Part 2: Access Protected Resources