NextJs

1. **DIFFERENCE BETWEEN REACT.JS AND NEXT.JS**

So React.js is actually a frontend JavaScript library, while Next.js is a full-fledged React framework built on top of it.

- ⚛️ **React.js**
  - It's mainly used for building **client-side** user interfaces.
  - With React, we have to handle everything ourselves only, be it, **routing** (using `react-router`), **state management**, **API calls**, and so on.
- 🚀 **Next.js**
  - It's like React but with superpowers.
  - We get **file-based routing**, **server-side rendering (SSR)**, **API routes**, **image optimization**, and even **SEO benefits**.
  - We can do **full-stack stuff** with Next.js since it also supports backend logic via API routes.

  So yeah, **React gives us the flexibility**, but we've gotta build a lot of things ourselves only. **Next.js gives us a full-stack experience** with better performance and SEO.

2. **DIFFERENCE BETWEEN LIBRARY AND FRAMEWORK?**

The main difference comes down to control:
📚 **Library**

- A library is like a **toolbox**. We call it when we need it.
- For example: **React** is a library where we decide how to structure our app, handle routing, etc.
  🏗️ **Framework**
- A framework is like a **blueprint**. It calls us. We're following its rules.
- For example: **Next.js** is a framework—it gives us routing, structure, rendering, backend logic… the whole flow.

So yeah, React gives you flexibility, but with Next.js (being a framework), you get structure and features out of the box

3. **WHAT IS SERVER SIDE RENDERING & CLIENT SIDE RENDERING IN NEXTJS? DOESN'T REACT HAVE THAT FEATURE?**

Yeah, so in **Next.js**, we get to decide how we want to render a page—either on the **server** or in the **browser**.

🖥️ **Server-Side Rendering (SSR)**

- In Server-Side Rendering, a Page is generated **on the server every time a request comes in**.
- It's useful for **dynamic data** and for boosting SEO, since search engines can crawl full HTML content.
- The server sends back a **complete HTML page**, so the user sees something immediately.
  🌐 **Client-Side Rendering (CSR)**
- In Client-Side Rendering, Page is rendered **inside the browser** after JavaScript loads.
- Initially, the user sees a loading spinner.
- It's good for dashboards, internal apps, or anything where SEO isn't a big deal.

🔁 **So does React have SSR?**

- **Not by default.** React itself only gives us the **UI rendering layer**. So if we want SSR in pure React, we would have to manually set up stuff like Node.js + Express + ReactDOMServer, which can get pretty messy.
- But **Next.js** handles all that boilerplate for us. So it's more powerful and production-ready.

4. **IF I HAVE ONLY CLIENT SIDE, THEN WHICH ONE SHOULD WE USE? REACT.JS OR NEXT.JS?**

If it's purely client-side, like a dashboard or an internal tool—then pure React.js is actually enough. We don't need SEO benefits there, because users are logging in

and interacting with the app *after* it loads.
But... if we want the flexibility to later add:

- SSR (server-side rendering) for performance,
- or server-side APIs (like auth, DB handling),
  then **Next.js can still be a smart choice**, even for client-heavy apps. We can choose to render things on the client using `"use client"`. So yeah:
- **React.js** is perfect if we only want a clean, simple SPA (Single Page App).
- **Next.js**: can be used if we want to expand or want better performance + structure

5. **WHAT IS SEO? HOW DOES IT WORK?**

SEO stands for Search Engine Optimization.
Basically, it's the practice of optimizing a website so that it ranks higher on search engines like Google. If an app is SEO-friendly, Google can **crawl its content**, understand it properly, and show it to users when they search for something related.
Here is the process of SEO in simple terms:

- **Search Engine Crawlers** (aka bots) visit our site.
- They read our page content, meta tags, links, titles, headings—all of that.
- Based on what they find, our page is indexed and shown to users searching similar stuff.
  Now, the better the page is optimized—for keywords, performance, mobile-friendliness, structured data—the higher it can appear in search results.
  ⚠️ But here's the catch with frontend applications:
- If the content is loaded **after** the page loads (like in React apps via API calls), crawlers might miss it—hurting your SEO.
- So if we want good SEO, we need the content to be present **right when the page loads**. That's where Server Side Rendering helps...