**NextJS theory**

1. NextJS is a React FRAMEWORK for production(a fullstack framework for React).
2. NextJS solves common problems and makes building React apps easier.
3. NextJS-Key features & Benefits: **Server-side Rendering**: -Automatic page pre-rendering:Great for SEO and inial load; -Blending client-seide and server-side:Fetch data on the server and render finished pages**; FILe-Based** **ROUTING:** -define pages and routes with files and folders instead of code;-Less code, less work, highly unserstandable; **FULLSTACK Capabilities**: - Easily add backend (server-side) code to your Next/React apps; - Storing data, getting data, authentication etc. can be added to your React projects.
4. NextJS allows us to determine WHEN a page should be pre-rendered.
5. File-based Routing instead of Code-based Routing: - Create React component files and let NextJS infer the routes from the folder structure into the special “/pages” folder.
6. With Next js we have page Pre-Rendering before the page is shown to the client. This is good for the SEO
7. Only the initial page is pre-rendered. Any other page accessed after have the same functionality as a React application.

Graphical user interface, application

Description automatically generated

1. Static Generation: -Pre-generate a page (with data prepared on the server-side) during buiold time; -Pages are prepared ahead to time and can be cached by the server / CDN serving the app. (the function for this is export await function getStaticProps(context)).
2. Next JS offers Incremental Static Generation: -It regenerate it on every requrest, at most every X seconds-> It server “old” page if re-generation is not needed yet/ It Generate, store and server “new” page otherwise.
3. In case there are dynamic paths, The NEXT JS have the Pre-Generated Path(Routes) function: - Dynamic pages ([id].js etc) don’t just need data, You also need to know which [id] values will be available; Multiple concrete [id] page instances (ex. Id=1, id=2 etc) are pre-generated. Function for this (export async function getStaticPaths(){})
4. The pages are pre-fetched with the dynamic generated pages.
5. Server-side Rendering: Sometimes you need to pre-render for every request OR you need access to the request object (ex. For cookies); NextJS allows you to run “real server-side” as well. Function => (export async function getServerSideProps())
6. You don’t need both getServerSideProps and getStaticProps, they usually have the same functionality but they are run at different times.
7. The server-side rendering function is run every time a request reach the server.
8. It is almost the same, but it has different timing, and different access to the context. The server side rendering has access to request and response and they can be manipulated.
9. Client-side Data fetching: Some data doesn’t need to be pre-rendered(Data changing with high frequency(stock data); Highly user-specific data(ex. Last orders in an online shop); Partial data(ex. Data that’s only used on a part of an page).
10. Pre-fretching the data for page generation might not work or be required => Traditional clien-side data fetching (ex. UseEffect(0 with fetch() if fine).

Graphical user interface, text

Description automatically generated

1. Custom hooks: SWR <https://swr.vercel.app/>. State while revalidate hook for fetching data with caching, errors …
2. Deploying options: 