**SERVER\_SIDE RENDERING**

Server-side rendering (SSR) is a technique that allows you to generate the HTML of your pages on the server, instead of on the browser. This can improve the performance, SEO, and user experience of your web application.

Some of the advantages of SSR are:

* Faster initial loading: The browser receives the pre-rendered HTML and can display the content without waiting for JavaScript to execute.
* Better SEO: Search engines can crawl and index the HTML content more easily, which can improve the ranking and visibility of your pages.
* Enhanced accessibility: Users with slow or unreliable internet connections, or with JavaScript disabled, can still access your pages and view the content.

Next.js is a React framework that supports SSR out of the box. You can use Next.js to create dynamic web applications that leverage the benefits of SSR.

To implement SSR with Next.js, you need to use the getServerSideProps function in your page components. This function runs on the server on every request and returns the props for your page. You can use this function to fetch data from external sources, such as APIs, databases, or files, and pass it to your page component.

For example, suppose you have a page that displays the current date and time. You can use getServerSideProps to fetch the date and time from an API and render it on the server. Here is how the code would look like:

// pages/date.js

import React from 'react'

export default function Date({ date }) {

// Render the date prop

return <div>The current date and time is: {date}</div>

}

// This gets called on every request

export async function getServerSideProps() {

// Fetch data from external API

const res = await fetch('https://worldtimeapi.org/api/ip')

const data = await res.json()

// Pass data to the page via props

return {

props: {

date: data.datetime

}

}

}

You can also use different data fetching strategies for SSR, depending on your needs. For example, you can use getStaticProps to pre-render your pages at build time, instead of on every request. This can improve the performance and reduce the load on your server, but it also means that your pages will not reflect the latest data changes.

You can also use getStaticPaths to pre-render only some of the pages with dynamic routes, and use fallback rendering for the rest. This can help you optimize the build time and the user experience, by generating the most popular or important pages ahead of time, and rendering the others on demand.