Next JS Notes

Code with Harry

* Getting Staarted
* Head components: It is useful for SEO purpose
* Script Component: If we want to import any 3rd party scripts such as google analytics and advertisement, then we can add scripts dynamically. It has some properties among them "strategy" property is used most often.

<script stratgey="lazyOnload">. Lazy-load renders the script in web page when all of the pages component is rendered properly

* Image Component : The **<Image>** component is separately created in Next.js to provide optimized image loading and better performance for images on websites. It facilitates responsiveness, automatic optimization.
* Lazy-loading : The **<Image>** component in Next.js supports lazy loading, which means that images are loaded only when they are about to appear in the user's viewport. This can significantly improve the performance of

pages that contain a large number of images.

**Dyanamic Routing:**

In Next.js, dynamic routing refers to the ability to create pages with dynamic URLs that can be based on user input or data from a database or API. This is achieved by defining a page file with a special naming convention that includes a parameter placeholder.

For example, if you create a file called **pages/posts/[id].js**, it would create a dynamic route for the "posts" section of your website. The **[id]** part of the filename represents a dynamic parameter that can be any value, such as a post ID, a slug, or any other identifier.

For instance, a URL like **https://example.com/posts/123** would match the **pages/posts/[id].js** file and pass the value "123" as a prop to the component. The component can then use this value to fetch data from a database or API and display the corresponding content on the page

**Link Component In Next JS**

In Next.js, the **Link** component is a built-in component that allows you to create client-side links between pages in your application, without triggering a full page reload.

**Applying CSS In Tailwind**

**CSS Module**

Next.js has built-in support for CSS Modules using the .module.css extension. Suppose if we have a reusable Button Component in components/Button folder.

We can create a module css file component/Button.module.css with following content for example

.error { color: white; background-color: red;}

Then, create components/Button.js, importing and using the above CSS file:

import styles from './Button.module.css';

export function Button() {

return (

<button

type="button"

className={styles.error}

>

Destroy

</button>

);

}

**Global CSS file**

Global CSS can be used by importin in the \_app.js file. Global CSS refers to CSS styles that are applied to an entire website or application, rather than specific components or elements.

**Styled JSX**

It allows us to write component-level css styles directly using a special syntax

function MyComponent() {

return (

<div>

<h1 className="title">Hello, world!</h1>

<style jsx>{`

.title {

font-size: 2rem;

text-align: center;

}

`}</style>

</div>

);}

Note : The scope of Styled jsx is only component-level however if we try to make it global, we can we just need to add ‘global’ in the syntax

<style jsx global> …</Style>

**Displaying Navbar In all Pages**

Suppose we have Home, About, Contact Pages in our website. We Certainly want to display some components like Navbar, Footer in all of our pages. In order to do that we have to include Navbar Component in our \_app.js

import Navbar from './Navbar'

*export* default function App({ Component, pageProps }) {

  return(

    <>

    <Navbar/>

    <Component {...pageProps} />

    </>

  )

}

**API Routes In Next JS**

API routes in Next.js are server-side endpoints that allow you to create APIs that can be called from your client-side JavaScript code. To create an API route in Next.js, you simply create a file in your project's "pages/api" directory.

export default function handler(req, res)

{ res.status(200).json({ name: 'John Doe' });}

This is an example of an API route in Next.js. The code exports a default function called **handler**, which takes two parameters: **req** and **res**.

**req** stands for request and **res** stands for response. When a client makes a request to this API route, the **handler** function is called with the incoming request object **req**, and the response object **res**.

The **handler** function is responsible for processing the incoming request and returning a response to the client. In this case, the function simply returns a JSON object with a status code of 200 and a single property **name** with the value of "John Doe".