**General Informations**

Next.js is an open-source web development framework created by the private company Vercel providing React-based web applications with server-side rendering and static website generation.

OR

Next.js is a popular open-source framework for building web applications and websites with React. It is based on the Node.js runtime and provides a streamlined development experience for creating server-rendered or statically generated React applications.

OR

Next.js is a React framework that gives you building blocks to create web applications. By framework, we mean Next.js handles the tooling and configuration needed for React, and provides additional structure, features, and optimizations for your application.

**Why we use Next.js?**

Next.js is widely used in the web development community due to its developer-friendly features, performance optimizations, and the ability to build modern web applications with ease. It has gained popularity for building static websites, blogs, e-commerce platforms, and even larger-scale applications.

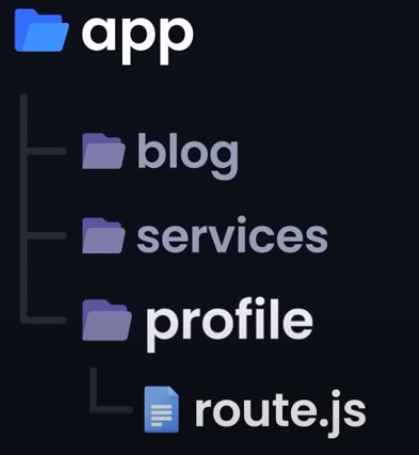
**What does Next.js have that React doesn’t Have?**Ans:

* Next.js simplifies the development process.
* On top of that it optimizes your web apps.

It does that through its primary features.

**Primary** **Features of Next.js:**

Here are some key features and concepts of Next.js:  
  
The primary distinction between react.js & next.js lies in how they handle rendering.

1. **Server-side rendering (SSR)**: React.js renders user interface on the client side while Next.js ***allows you to render React components on the server*** i.e. (Server-Side Rendering) and send HTML to the client, which improves performance and enables search engine optimization (SEO). This means that your web pages can be pre-rendered on the server and delivered as fully-rendered HTML to the browser, providing a faster initial load time and better SEO visibility.  
     
   However, Next.js offers flexibility in rendering options you can choose to render the UI on the client side or the server side according to your needs.
2. **Client-side rendering (CSR)**: Next.js can also handle client-side rendering when needed. This allows you to build dynamic, interactive web applications by fetching data on the client side and updating the UI without reloading the entire page.
3. **Search Engine Optimization (SEO):** SEO is crucial for optimizing a website’s visibility and ranking in search engine results.  **Benefits due to higher search result rankings:***-- Increased Organic traffic  
   -- Enhanced user experience  
   -- Credibility & trustworthiness  
   -- competitive advantage*   
     
   Prioritizing SEO can greatly impact the success of your website and its online presence. Search Engine Crawlers face difficulties indexing Pages dynamically rendered on the client side as a result the SEO performance of such pages may suffer. As search engines may not fully comprehend their content and rank them appropriately.  
   By utilizing next.js this issue is resolved by ***sending pre-rendered code directly to the client***.  
   This enables: Easy crawling and indexing by search engines leading to the improved SEO.
4. **Routing**: Next.js provides a simple and intuitive routing system. You can define dynamic routes, nested routes, and catch-all routes easily, making it straightforward to create complex applications with multiple pages.
5. **Ability to create Full-Stack Application**: Form Next.js version 9 developers behind Next.js introduced a new feature called **API Routes**. Enabling the creation of *serverless functions* to handle API requests. Serverless APIs in Next.js are a way of creating API endpoints without the need for a traditional server.  
   It allows us to build and deploy APIs:  
   -- Without managing server infrastructure Or  
   -- Worrying about scaling their server as traffic increases.  
     
   With this feature, we can create API endpoints by simply creating a file called route.js in a specific folder within the app directory.  
   

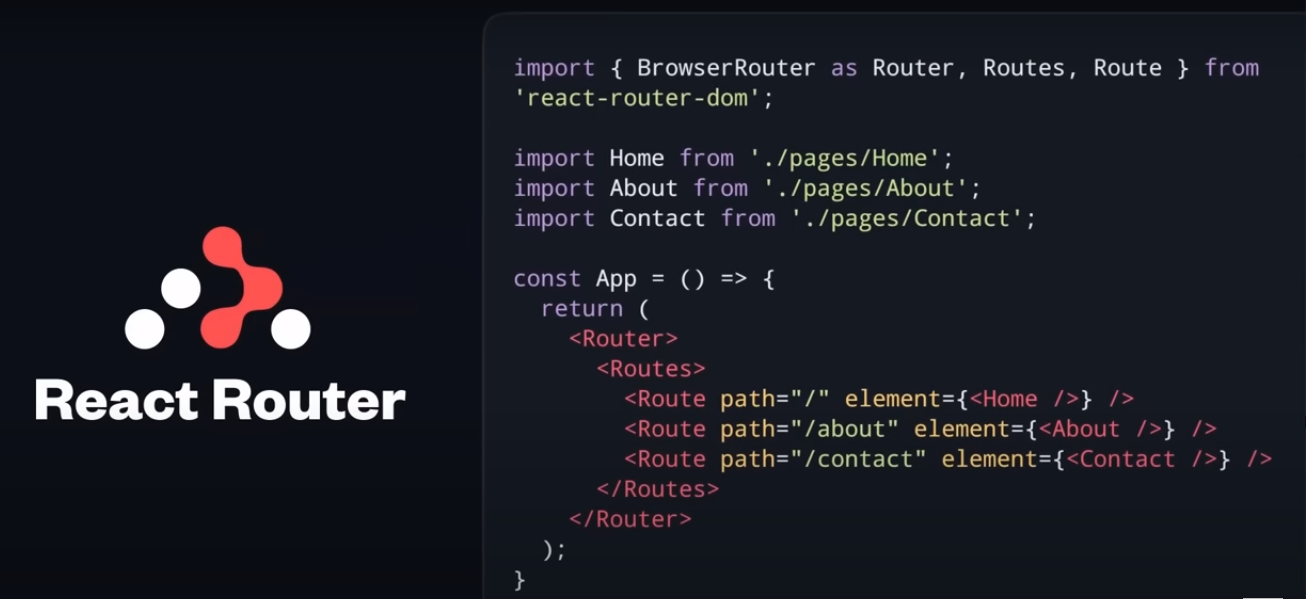
This file in any route segment of the directly corresponds to that route API endpoint.

1. **Automatic code splitting**: Code splitting is a technique that breaks down large bundles of JavaScript code into smaller, more manageable chunks that can be loaded as needed.   
   Next.js ***automatically splits your JavaScript code into smaller chunks***, which are loaded only when needed. This improves initial load times and ensures that users only download the code they actually need for a particular page.  
     
   Although we can achieve code splitting in react but the process is manually. We’ve to do lots of configuration as your application grows.  
     
   For ex:  
   We need to use lazy() function from react to dynamically import the about component only when it’s needed.  
   We also use the Suspense component to show a fallback UI when the component is being loaded.

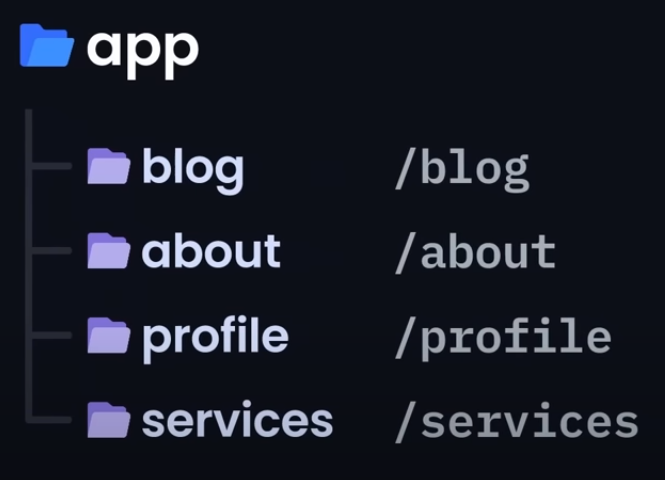


But in Next.js this process is entirely automatic. It uses automatic code splitting by default to split pages into separate chunks resulting in Faster subsequent page navigations.

1. **API routes**: Next.js includes an API routes feature that allows you to define serverless endpoints within your application. These endpoints can handle HTTP requests and perform server-side logic, providing a convenient way to build backend functionality alongside your frontend code.
2. **Static site generation (SSG)**: Next.js supports generating static HTML files at build time, allowing you to pre-render pages in advance and serve them directly from a CDN (Content Delivery Network). This approach is ideal for websites with content that doesn't change frequently, providing fast and scalable performance.
3. **TypeScript support**: Next.js has built-in TypeScript support, enabling you to write type-safe code and benefit from enhanced tooling and developer productivity.

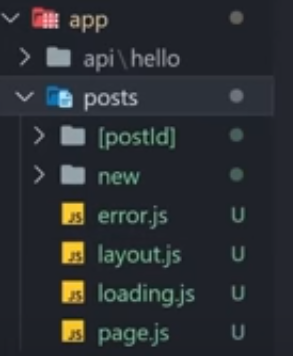
**How do we create different page routes in React & Next.js?  
Ans**.:   
In React:  
We have to install an additional package called React Router Dom and then create rotes in React.  


In Next.js:  
Next.js uses file-based routing system which means that routing is handled by the file system.   
Each Folder inside app directory becomes a rout.

  
If we have folder name about, we can access about page by:  
[*http://localhost:3000/about*](http://localhost:3000/about)No need for external packages or complex configurations.

**Note:-**

* In Next.Js Project, routes will be created as per Folder & files name. These names are case sensitive in the routes, so make sure all folder & file name in lower-case.  
  Ex:  
  Let say we want to create an api to update task in todo app, where id is going to take from the request as query. So id would be dynamic. To make id as dynamic route, we need to make id file, where its file name would look like [id].  
  Then it will access like, http://localhost/api/task/[id]  
  Where in the place of [id], would be dynamic id like:  
  6482f221a904344b8cd8d8d4
* Client Side Rendering happens on the client’s device or the browser when a user requests a web page the server sends a basic HTML document and JavaScript code the browser then downloads and executes the JavaScript code which leads to the rendering of components and finally the display of the website.  
    
  Server-side Rendering involves rendering the web page on the server before transmitting it to the client’s device. When a user requests a page the server processes the request and renders the components on the server side, the server then sends back the fully rendered HTML to the client’s browser enabling immediate display.
* Next.js is an extension of React that streamlines the development process by automating several functions Allowing developers to focus on what they do best writing React code.
* Error handling and show loading is easy in Next.js. We just need to add convention. For error handling we need to create error.js & for loading loading.js file in any sub-folder as per requirement within App directory. Whenever error comes, error.js file will automatically execute.   
  Error components must be client components.

****