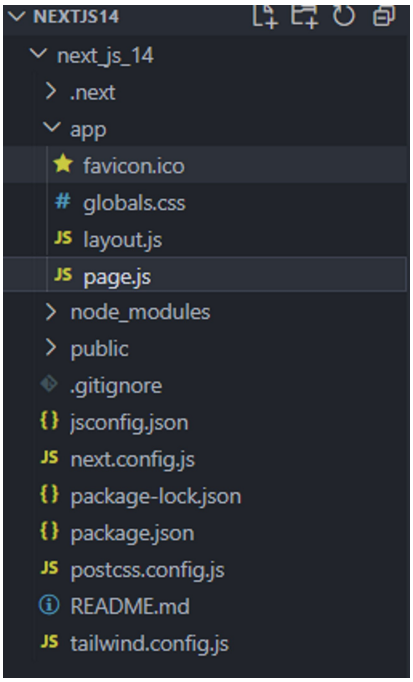
Getting started

Monday, December 18, 2023 3:53 PM

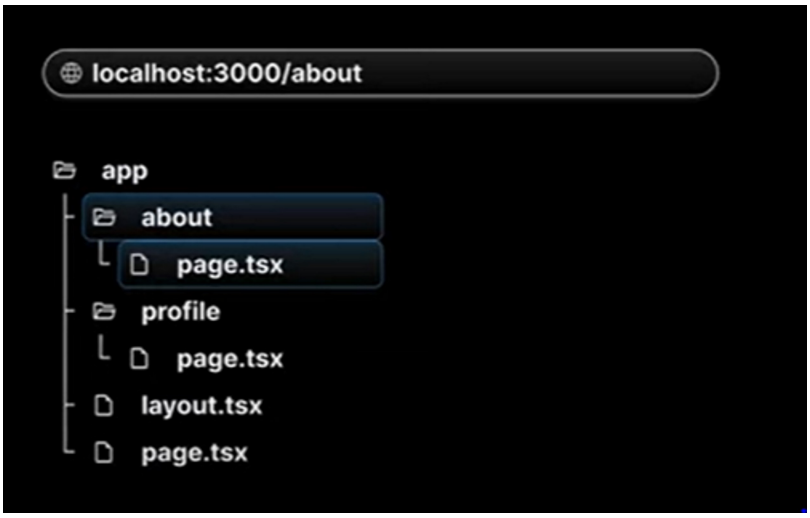
- We can get a next JS project installed with the command - npx create-next-app@latest
- Start the development server using the command- npm run dev
- Folder structure-
- 

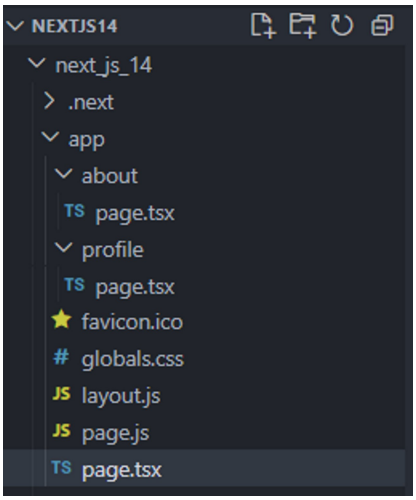
- In next JS all components are server components by default. They all have the ability to run tasks like reading files or fetching data from a server.
- But we cannot add hooks in them.

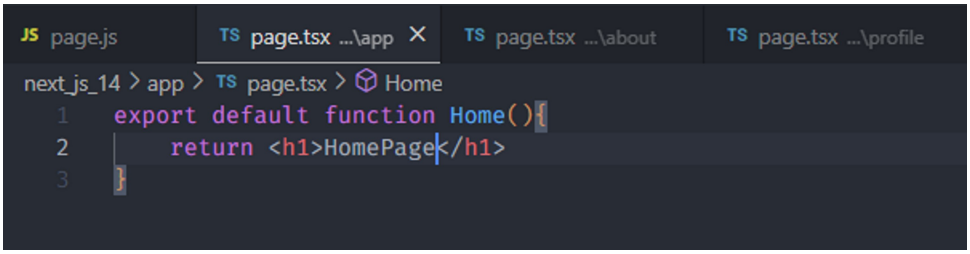
Routing

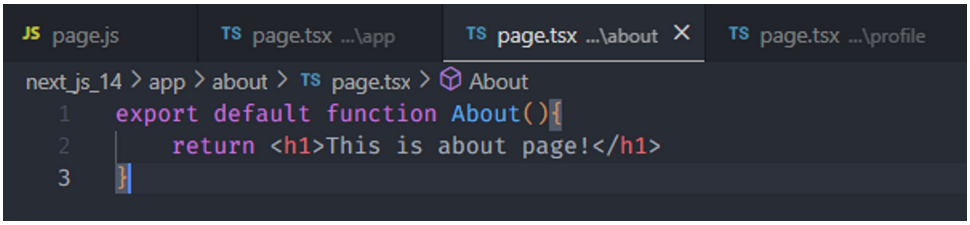
Friday, December 22, 2023 11:34 AM

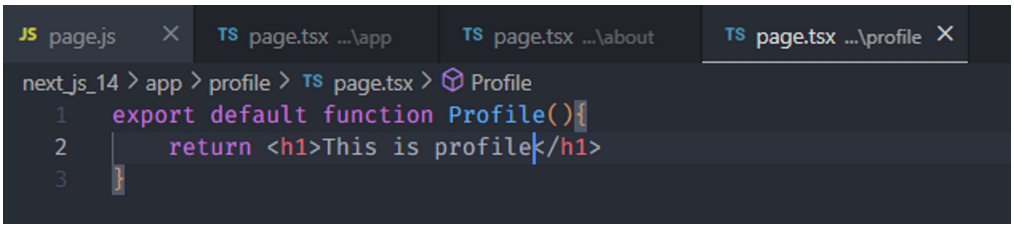
- All routes must be placed inside the app folder
- Every file which corresponds to a route must be name page.js or page.tsx depending on the fact that it is a simple JS file or a typescript file
- Each folder corresponds to a path segment in the browser URL
- We create code folders according to our requirements and the change in the URL changes the path of the file being rendered.

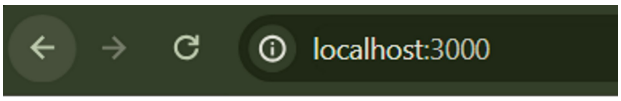




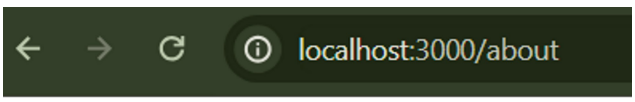




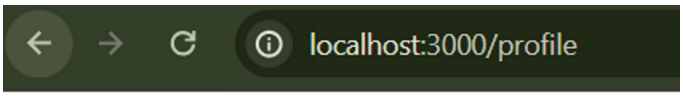




HomePage



This is about page!

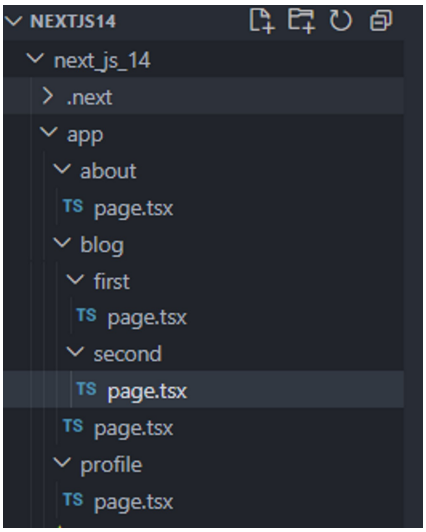


This is profile

Nested Routes

Friday, December 22, 2023 12:33 PM

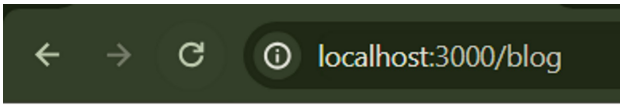
In case of nested folder structure , we just have to create a sub folder structure following the same file naming convention as before. SO by changing the URL we can access the sub files also along with their content.



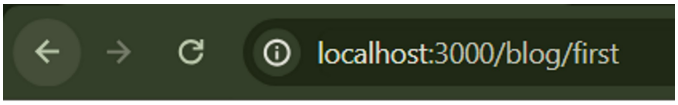

localhost:3000/blog
app
blog
first
page.tsx
second
page.tsx
page.tsx
layout.tsx
page.tsx


localhost:3000/blog/first
app
blog
first
page.tsx
second
page.tsx
page.tsx
layout.tsx
page.tsx


localhost:3000/blog/second
app
blog
first
page.tsx
second
page.tsx
page.tsx
layout.tsx
page.tsx



Blog page!



This is first page!

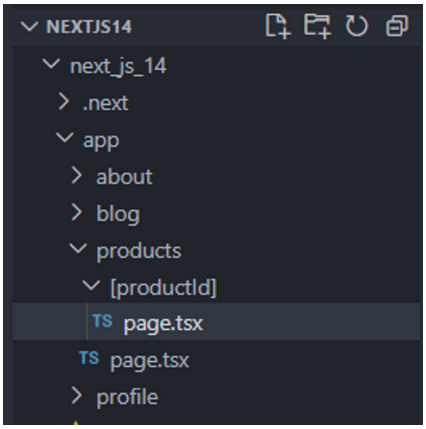
This is second page!

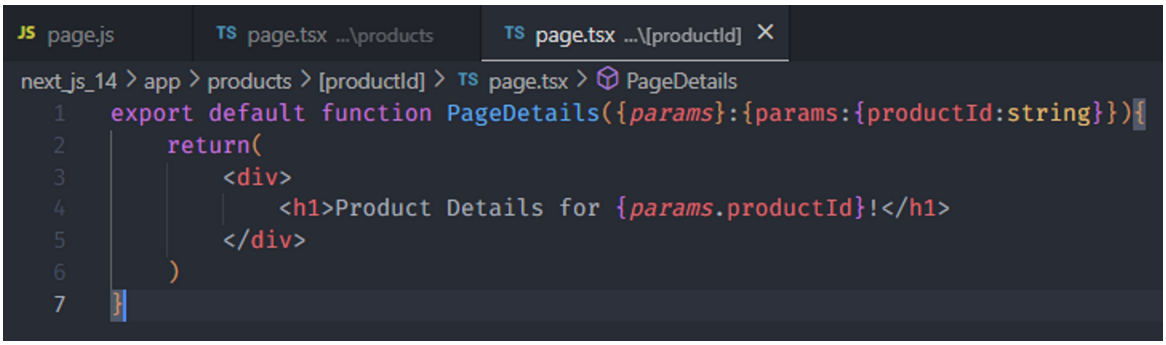
Dynamic routes

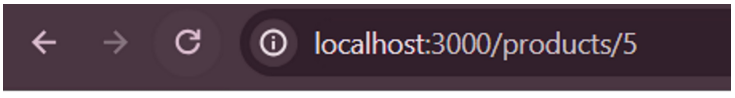
Friday, December 22, 2023 1:09 PM

This can be the case which is seen in many e-commerce websites . We change the data of the page according to the id of the product for which the data is required. We cannot make separate details page for each products so we have to move to the concept of dynamic routing.

For example in case we have to make a products detail page for each product , we need to make a unique folder name enclosed within square brackets.

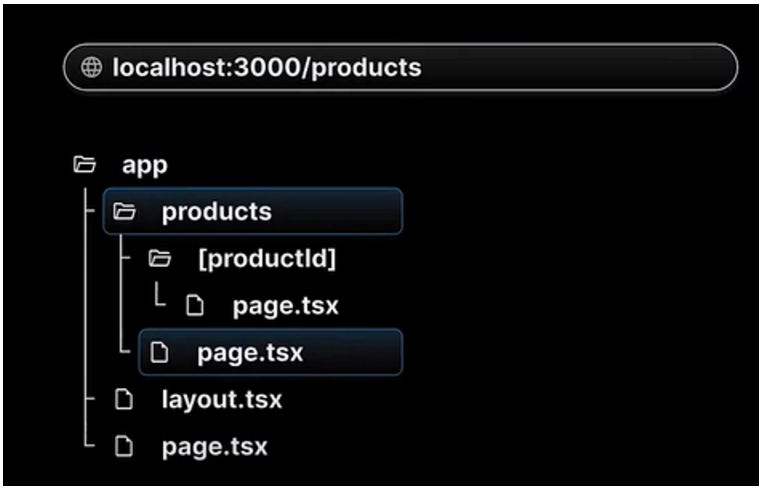


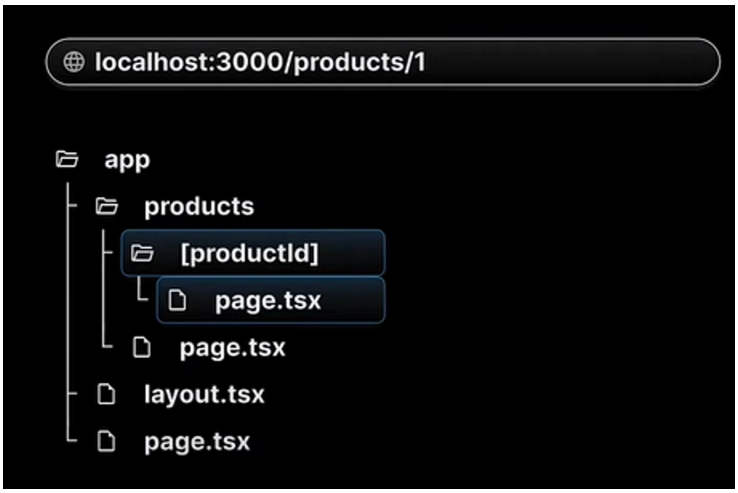




Product Details for 5!

Path flow-



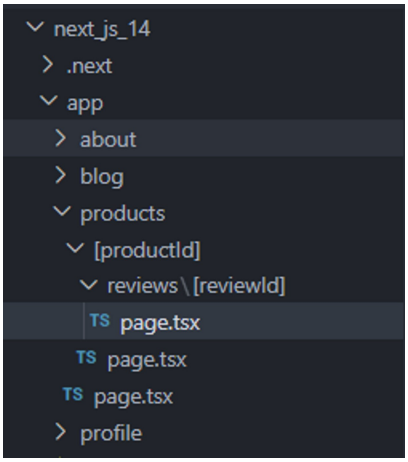


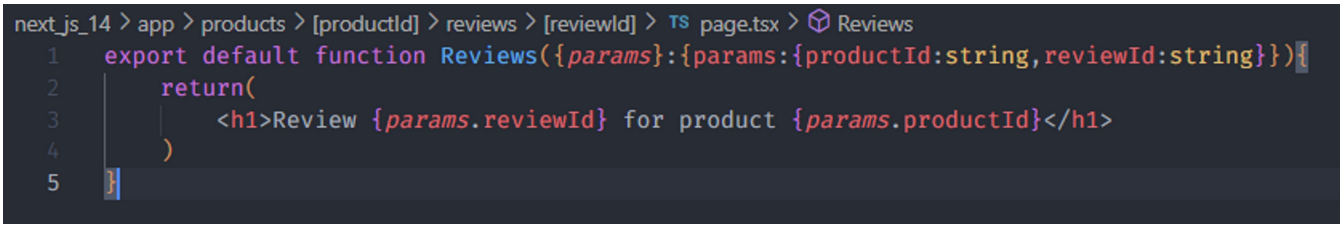
So whenever the path of the page is changed according to the productid , the id is fetched in the file and then relevant data is shown through the DB or whatever the data source is.

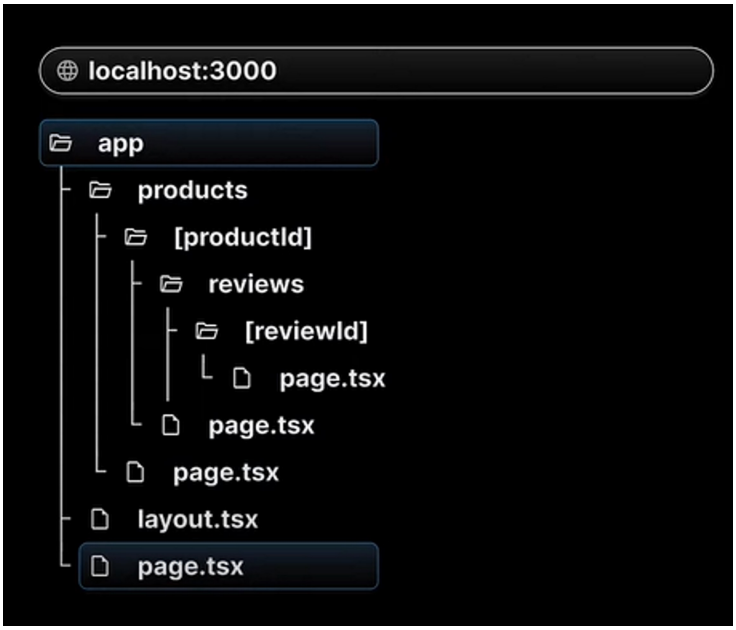
Nested Dynamic Routes

Friday, December 22, 2023 5:57 PM

What about the case if the nested routing needs more nesting? Like for a specific product we need to have specific reviews also? Then comes the case of nested dynamic routing.

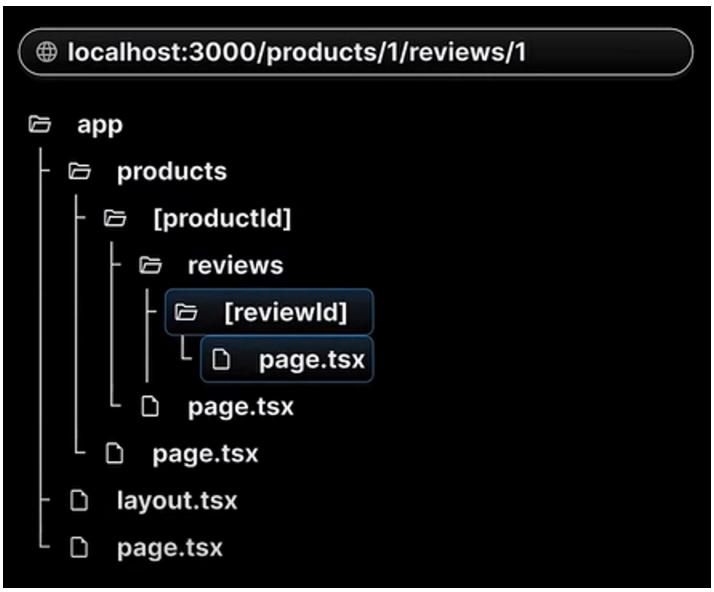






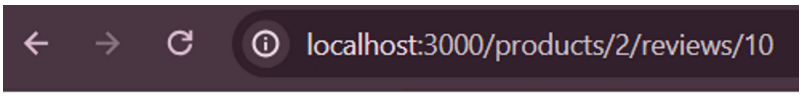

localhost:3000/products
app
products
[productId]
reviews
[reviewId]
page.tsx
page.tsx
page.tsx
layout.tsx
page.tsx


localhost:3000/products/1
app
products
[productId]
reviews
[reviewId]
page.tsx
page.tsx
page.tsx
layout.tsx
page.tsx

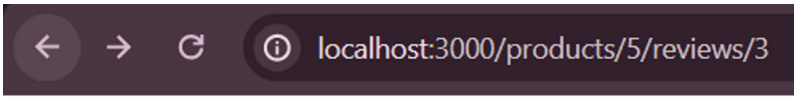

localhost:3000/products/1/reviews/1
app
products
[productId]
reviews
[reviewId]
page.tsx
page.tsx
page.tsx
layout.tsx
page.tsx

Working-

We navigate to the specific product through the means of dynamic routing.
After that we have a nested reviews folder in which the reviews layout will be saved. Inside it we have the reviewId folder in which the specific review layout will be saved and on the entering of the specific review the details will be shown to the user.



Review 10 for product 2

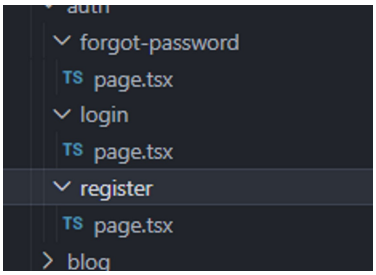


Review 3 for product 5

Route Groups

Tuesday, December 26, 2023 12:45 PM

Allows us to group our routes and project files without affecting the URL path structure.

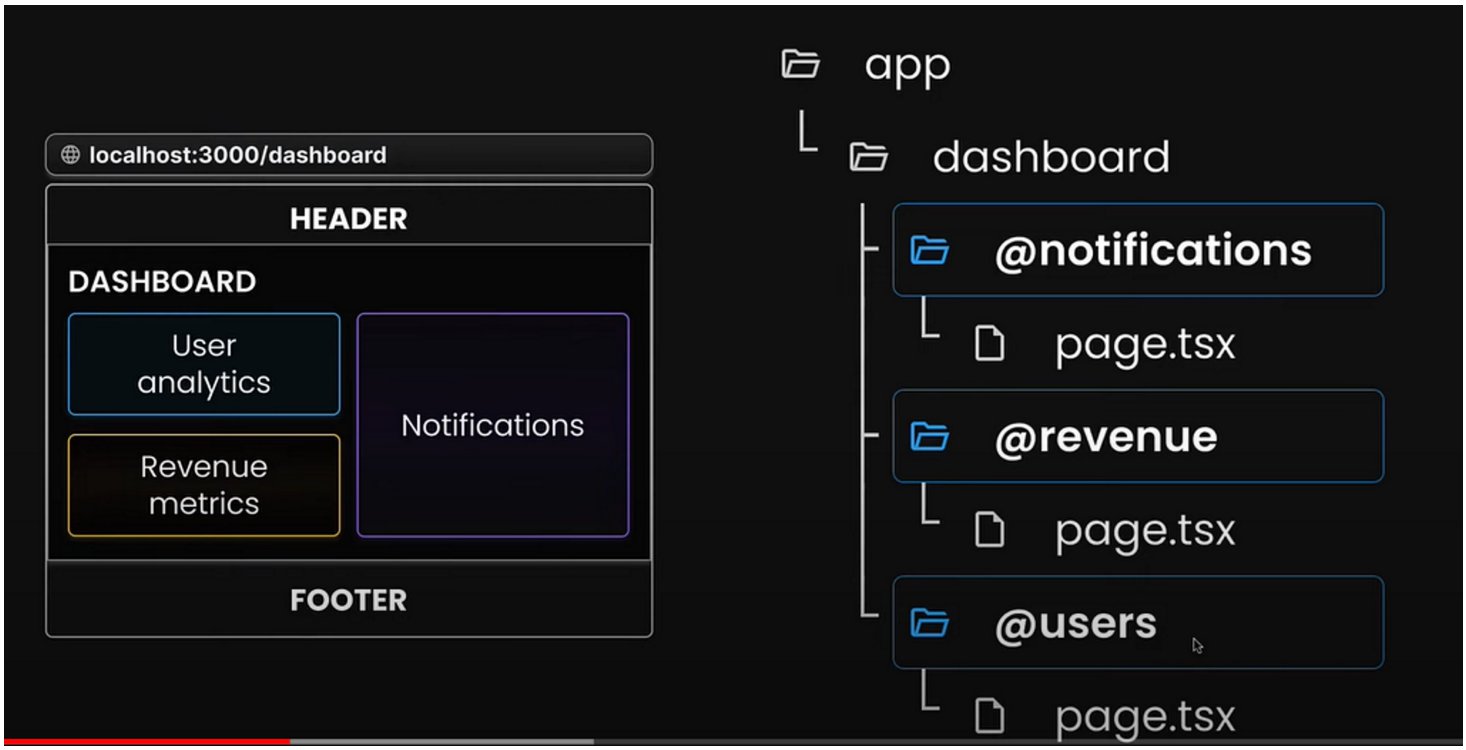


Parallel Routes

Wednesday, December 27, 2023 3:26 PM

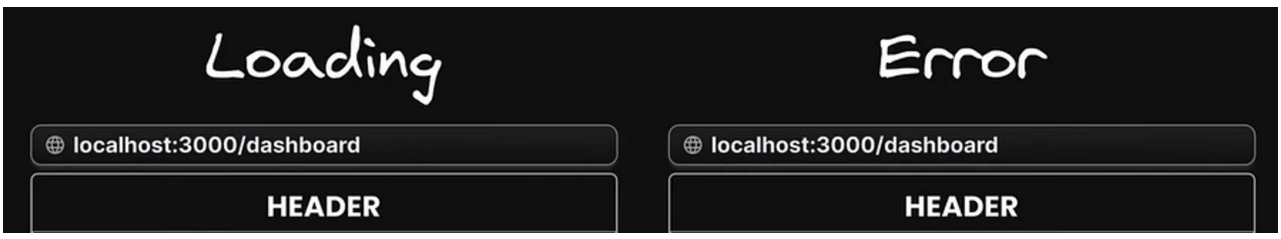
Parallel routing allows for the simultaneous rendering of multiple pages within the same layout.

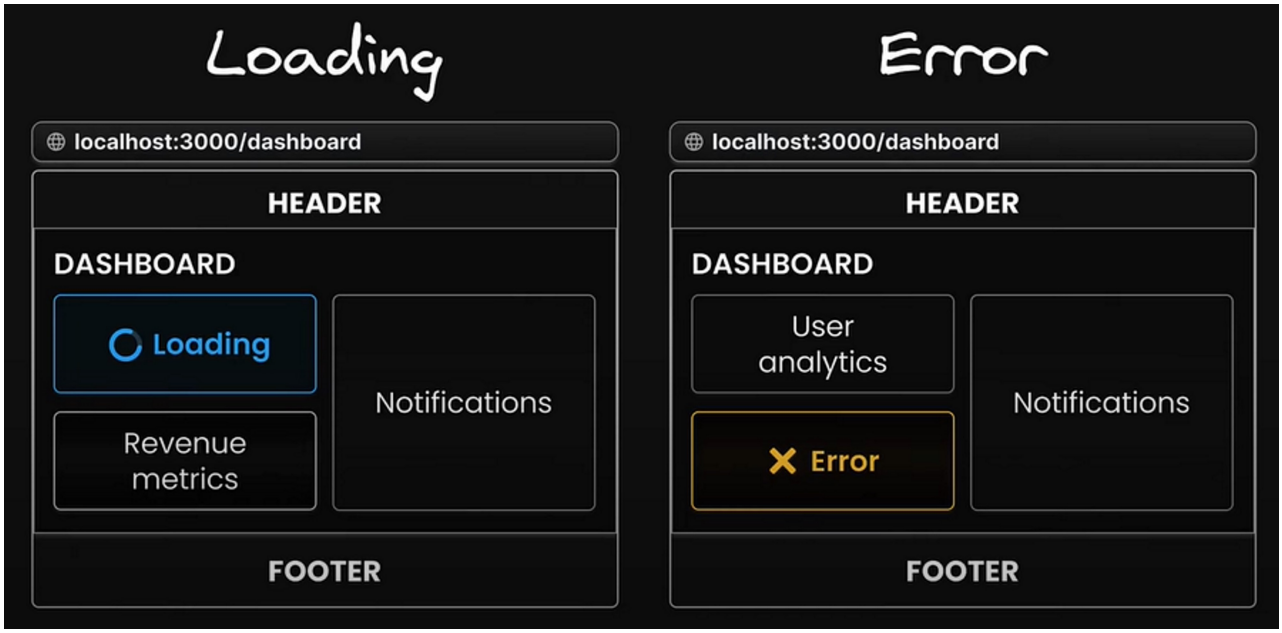
- Parallel routes are defined in Next JS using a feature called slots
- Slots help structure our content in modular fashion
- To define a slot, we use the `@folder` naming convention
- Each slot is passed as a prop to its corresponding layout.tsx file



Benefits of parallel routing-

- Being able to split a single layout into multiple slots making the code more manageable
- Independent route handling
- Sub Navigation
- Each slot can have its own loading and error handling states
- This granular control of section sis particularly profitable where different sections of the page load at different times or have different error handling situations



- 

Handling unmatched routes-

When we navigate through routes with the help of UI, the other components of the page layout like children etc remain unaffected

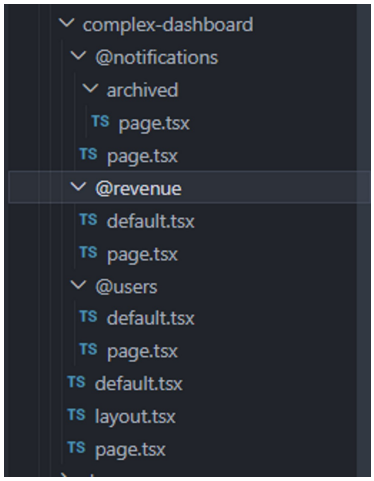
In case of page reload , next JS searches for default.tsx file within each unmatched slot. This file is critical since it provides the default content that next JS will render in the UI

If the default file is not present a 404 error page will be rendered.

Default.tsx file-

- It serves as a fallback to render content when the framework cannot retrive a slot's active state from the URL.
- Based on our requirements , we have the option of making the default page whatever our need is.

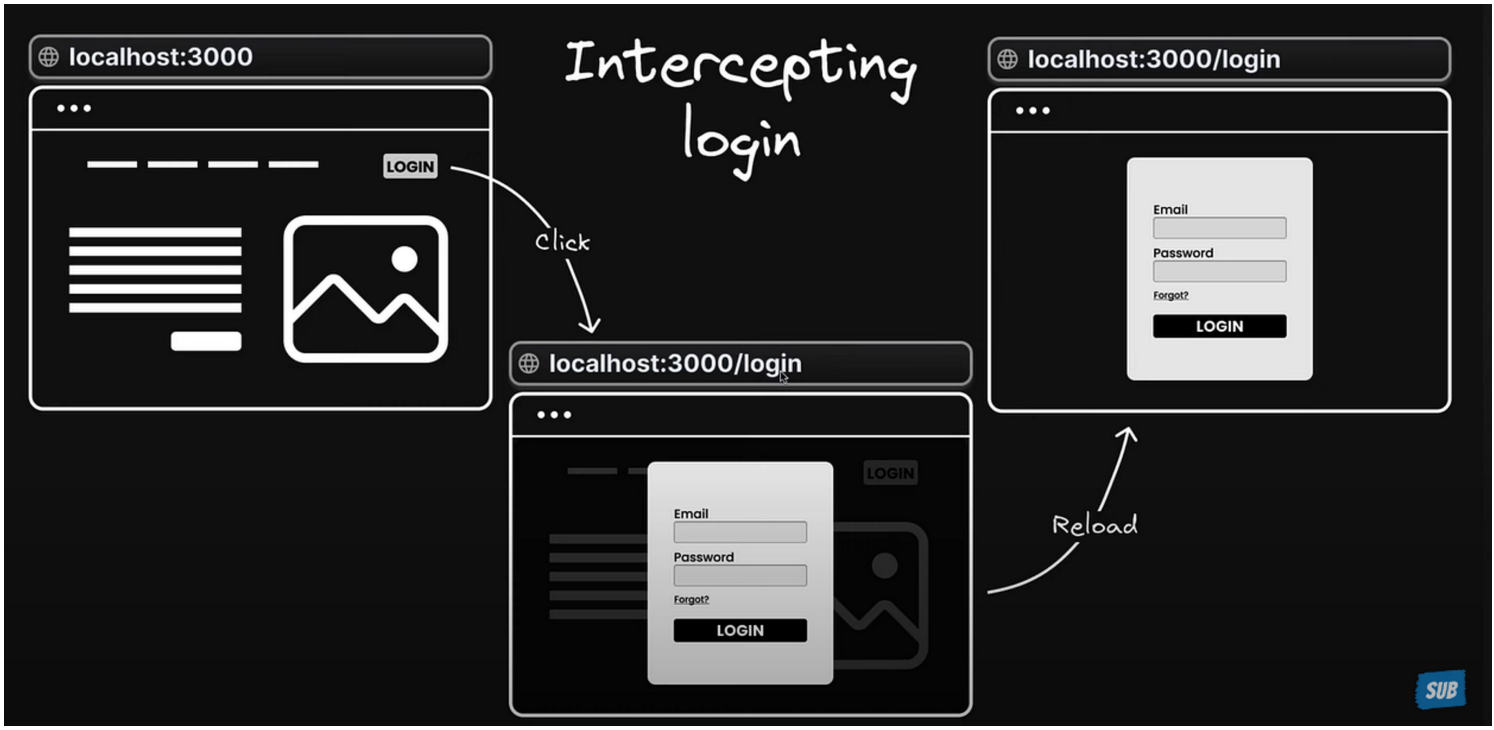
The default views should be at the same level as page.tsx


complex-dashboard
@notifications
archived
page.tsx
page.tsx
@revenue
default.tsx
page.tsx
@users
default.tsx
page.tsx
default.tsx
layout.tsx
page.tsx

Intercepting routes

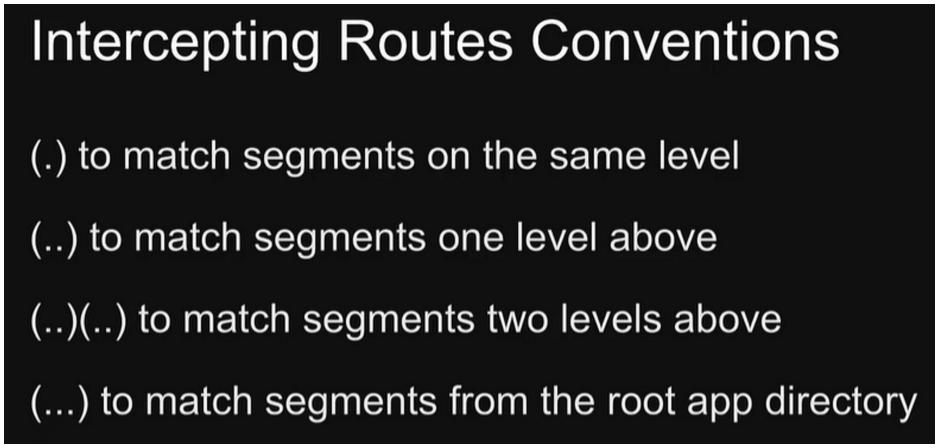
Wednesday, December 27, 2023 7:18 PM

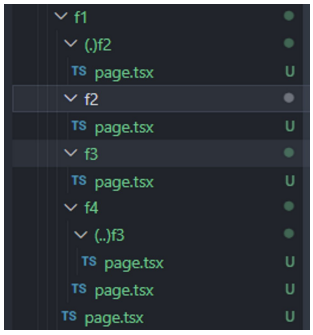
Intercepting routes allow us to intercept or stop having the default behavior to present an alternate view or component when navigating through the UI, while still preserving the intended route for scenarios like page reloads.



Like in this case , clicking on login button takes us to the login page. But by using intercepting routes we can get a login modal while still having the login route as the path. If the page is reloaded or someone access the page using the shared link, the whole login page is shown to the user.

So the idea is to display a different content on the page redirection but change the content when the page is reloaded.




f1
(.)f2
page.tsx U
f2
page.tsx U
f3
page.tsx U
f4
(..)f3
page.tsx U
page.tsx U
page.tsx U

Catch all path segments

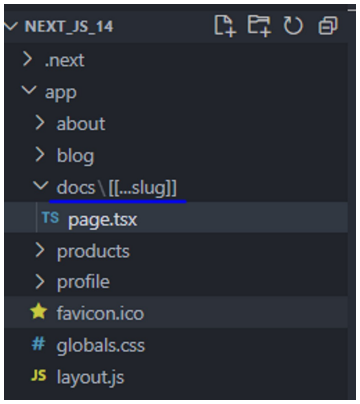
Friday, December 22, 2023 6:48 PM

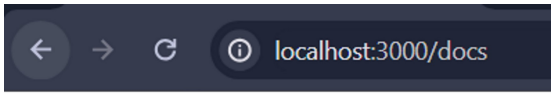
There can be a case where we need to have multiple sub pages inside a sub page.
Eg- localhost:3000/docs/feature1/concept1
localhost:3000/docs/feature1/concept2
localhost:3000/docs/feature2/concept1
localhost:3000/docs/feature2/concept2

If we have say 20 features and inside each feature we have 20 concepts , then we have to do routing for 400 pages. Making a separate page for each is out of the question. So we need to move to the concept of catching all path segments.

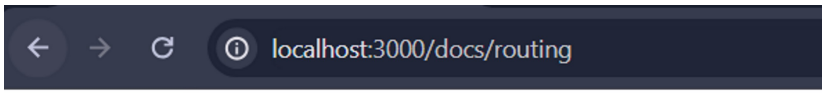
We can make folder structure like- 1[featureId]x 1[conceptId]

For each sub page we have to make a separate routing sub folder.

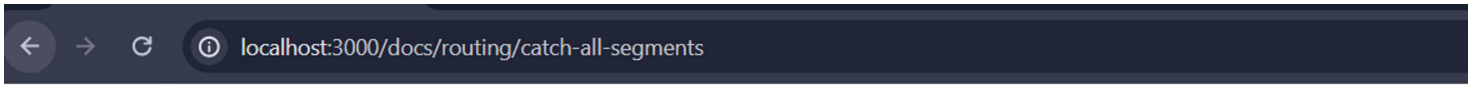




Docs home page!



Viewing docs for feature routing



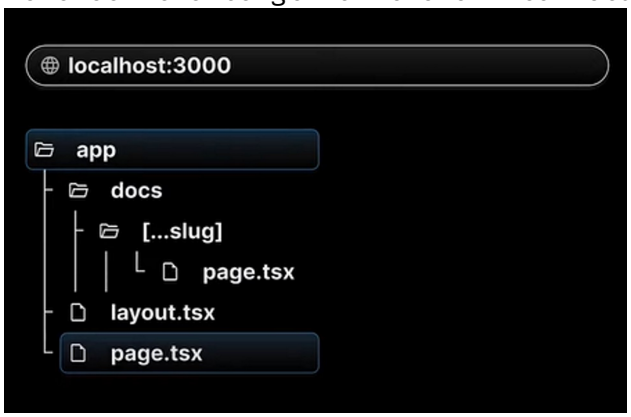
Viewing docs for feature routing and conceptcatch-all-segments

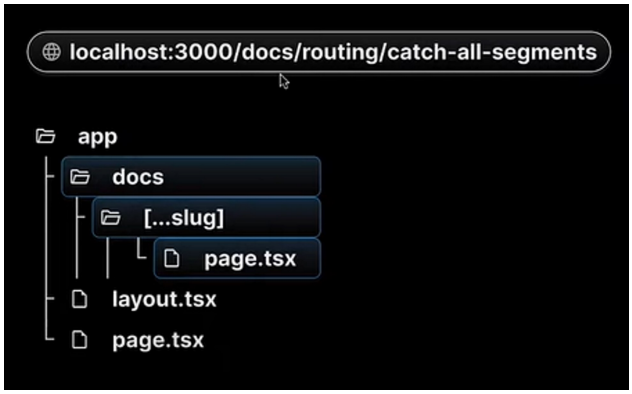
page.tsx | page.ts

app > docs > [[...slug]] > page.tsx > Docs

```tsx
export default function Docs({params}:{params:{slug:string[]}}){
    if(params.slug?.length==2){
     return(
         <h1>Viewing docs for feature {params.slug[0]} and concept{params.slug[1]}</h1>
     )
    }else if(params.slug?.length==1){
     return(
         <h1>Viewing docs for feature {params.slug[0]}</h1>
     )
    }
    return <h1>Docs home page!</h1>
}
```

We check the length of the URL to return different pages.

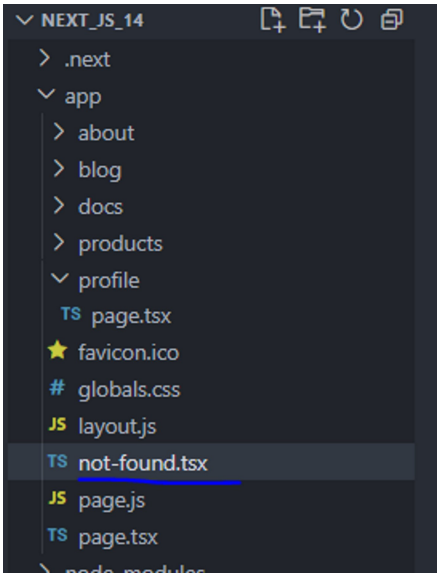


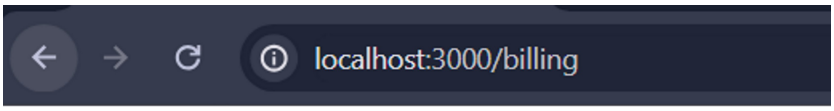


Not Found Page

Tuesday, December 26, 2023 11:40 AM

Sometimes we may need to create a custom 404 error page.

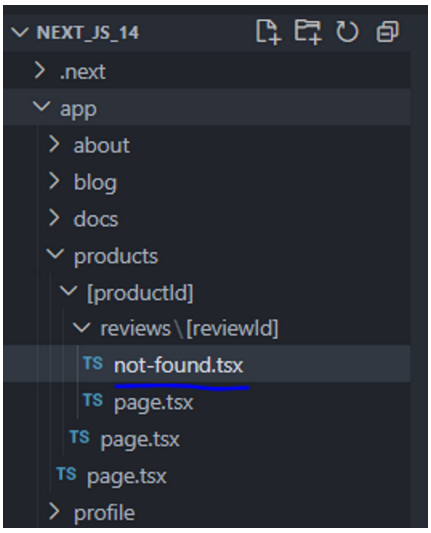




This is a custom error page!

But what if we have to programmatically render this error page? We can use a function from Next JS for this.

Let's say we cannot have reviews more than 1000 and we try to check for 1001



page.tsx ...\[[...slug]] | not-found.tsx | page.tsx ...\[reviewId] | page.ts

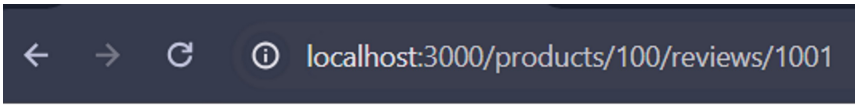
app > products > [productId] > reviews > [reviewId] > page.tsx > Reviews

```tsx
import { notFound } from "next/navigation"

export default function Reviews({params}:{params:{productId:string,reviewId:string}}){
    if (parseInt (params.reviewId)>1000){
        notFound();
    }
    return(
        <h1>Review {params.reviewId} for product {params.productId}</h1>
    )
}
```

app > products > [productId] > reviews > [reviewId] > not-found.tsx > NotFound

```tsx
export default function NotFound(){
    return(
        <div>
            <h2>This review cannot be found!</h2>
        </div>
    )
}
```



This review cannot be found!

Private folders

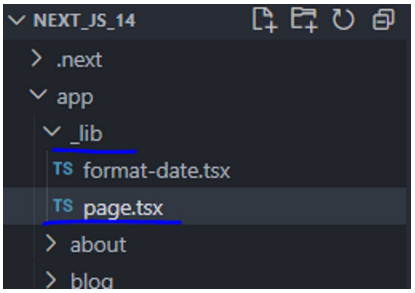
Tuesday, December 26, 2023 11:58 AM

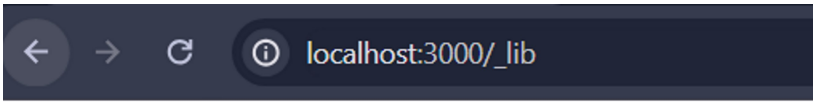
A private folder indicates that it is a private implementation and does not need to be considered in the normal routing segment. Its folders and sub folders are excluded from routing

Prefix the folder name with an underscore

We can use private folders for-

- For separating UI logic from routing logic
- For consistently organizaing internal files across the project
- For sorting and grouping files in code editors
- For avaoiding any potential naming conventions with the future Next JS file conventions





This is a custom error page!

Layouts

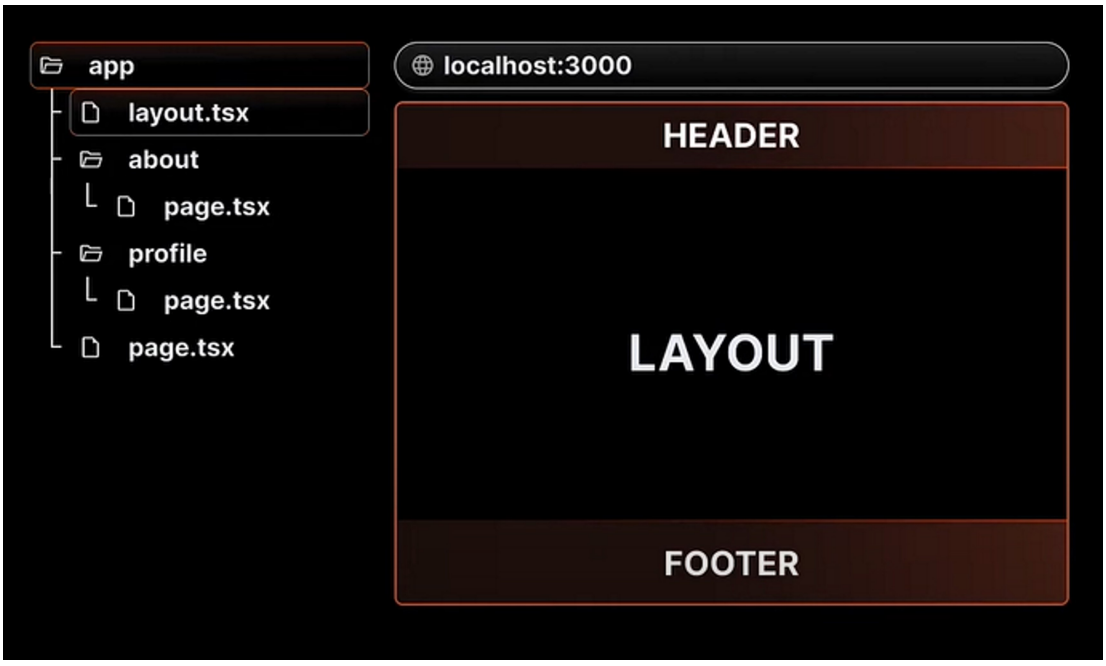
Tuesday, December 26, 2023 1:23 PM

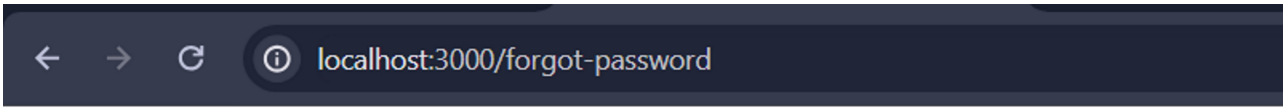
We can define a layout by default exporting a React component from a layout.js or layout.tsx file.

The component should accept a children prop that will be populated with a child page during rendering

app > JS layout.js > ...

```
1  import { Inter } from 'next/font/google'
2  import './globals.css'
3
4  const inter = Inter({ subsets: ['latin'] })
5
6  export const metadata = {
7    title: 'Create Next App',
8    description: 'Generated by create next app',
9  }
10
11 export default function RootLayout({ children }) {
12   return (
13     <html lang="en">
14       <body className={inter.className}>{children}</body>
15     </html>
16   )
17 }
18
```





Header

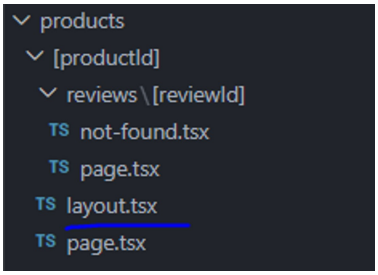
This is forgot password page!

Footer

Nested Layouts

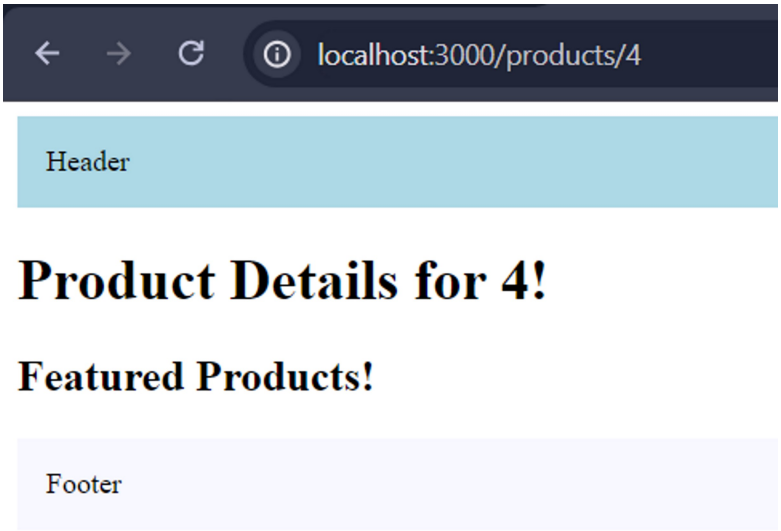
Tuesday, December 26, 2023 1:31 PM

We can define special layouts for each page by going into its routing folder and defining a layout file . The page will follow the layout with the data coming as props in the children value callout.



app > products > [productId] > layout.tsx > ProductDetailsLayout

```tsx
export default function ProductDetailsLayout({
    children,
}:{
    children:React.ReactNode;
}){
    return(
        <>
            {children}
            <h2>Featured Products!</h2>
        </>
    )
}
```

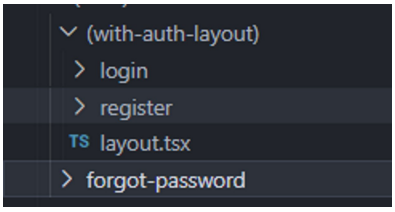


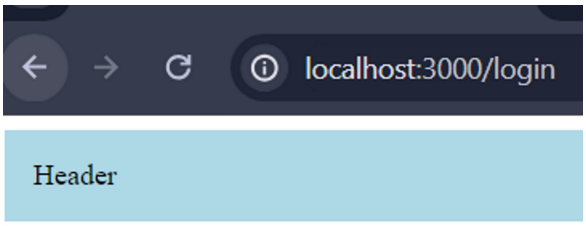

app
layout.tsx
products
[productId]
layout.tsx
page.tsx
page.tsx
page.tsx
localhost:3000/products/1
HEADER
Product 1
Featured products
FOOTER

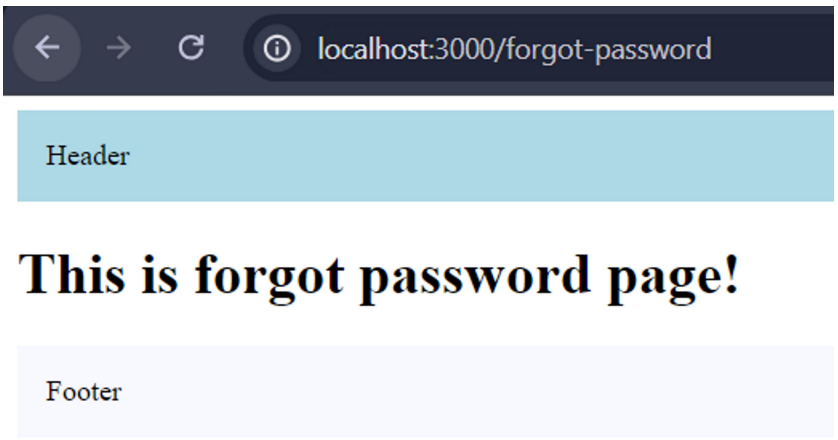
Route Group Layout

Tuesday, December 26, 2023 3:16 PM

We can define specific layouts for route groups also. Just make a new layout under layout.js/tsx file and then it will apply to the files inside that route group.







Routing Metadata

Tuesday, December 26, 2023 3:51 PM

Proper SEO is required to increase page visibility to the users.
Next JS introduced Metadata API which allows us to define metadata for each page.
Metadata ensures accurate and relevant information is displayed when your page is shared or indexed.

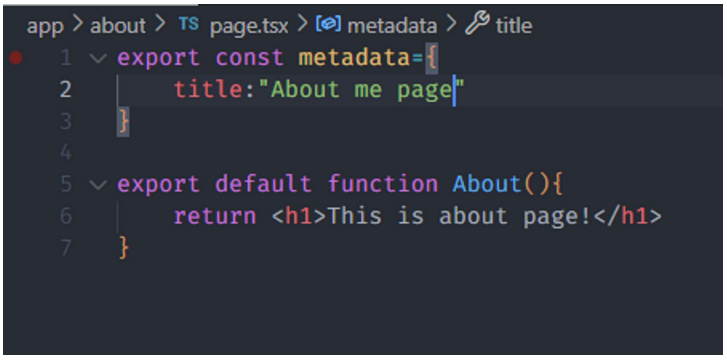
There are two methods to configure metadata-

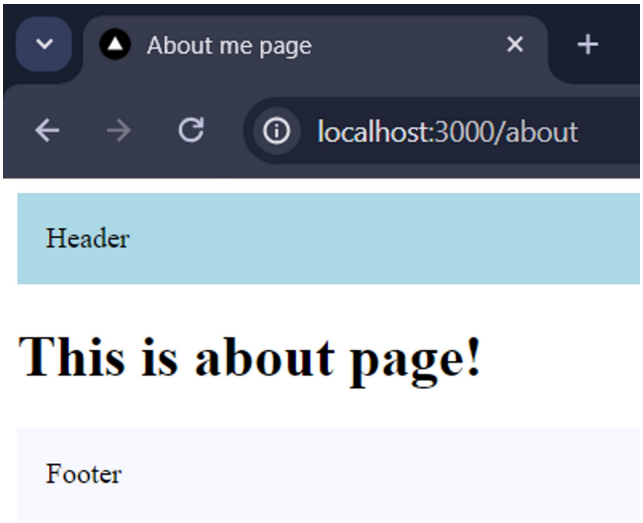
- Export a static metadata object
- Export a dynamic generateMetaData object

Metadata rules-

- It can be both applied to layout and page files. If defined to layout , it will apply to all the pages using that layout. And if applying to the page , it will specifically apply to that page.
- Its read from the root level to the final page level
- If there is multiple metadata defined in the same route, they are combined but page metadata will replace layout metadata if they have the same properties

Static Metadata export-





Here the two metadata are merged together where the deeper level metadata takes priority. So its changed with respect to the metadata defined inside the layout file.

Dynamic metadata export-

We need to get the attribute which will be dynamic in the page meta data.

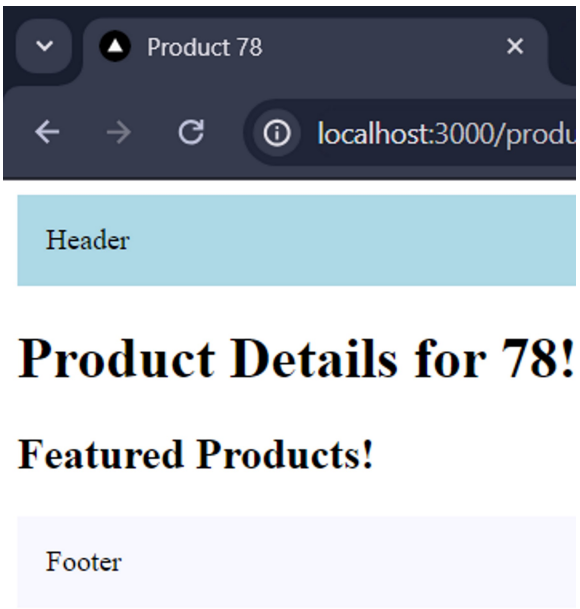
app > products > [productId] > page.tsx > generateMetadata > title

```tsx
import { Metadata } from "next";

type Props = {
  params: {
    productId: string;
  };
};

export const generateMetadata =async ({ params }: Props): Promise<Metadata> => {
    const title=await new Promise(resolve=>{
        setTimeout(()=>{
            resolve(`Product ${params.productId}`)
        })
    });
  return {
    title: `${title}`,
  };
};

export default function PageDetails({ params }: Props) {
  return (
    <div>
      <h1>Product Details for {params.productId}!</h1>
    </div>
  );
}
```



Here we have defined the props and the attribute type in the Props. We have a generateMetaData function and then we pass the params inside the function through which we are defining the dynamic part of the metadata.

Title metadata

Tuesday, December 26, 2023 7:30 PM

We have the option of providing object title through the meta data of a page . It accepts three values-

- Absolute
- Default
- Template

- Default - In this case we do not have a specific title defined for page, then the default value will be given to the title of the page

- Absolute-Through the absolute title we can set the title of all the page

- Template- In this case we define parts of the meta data. One part for defined by the actual page (denoted by %s) followed by some extra title data value as required. The title data provided is set inside the root file of the sub folder in the title attribute of metadata function of the type MetaData as imported.

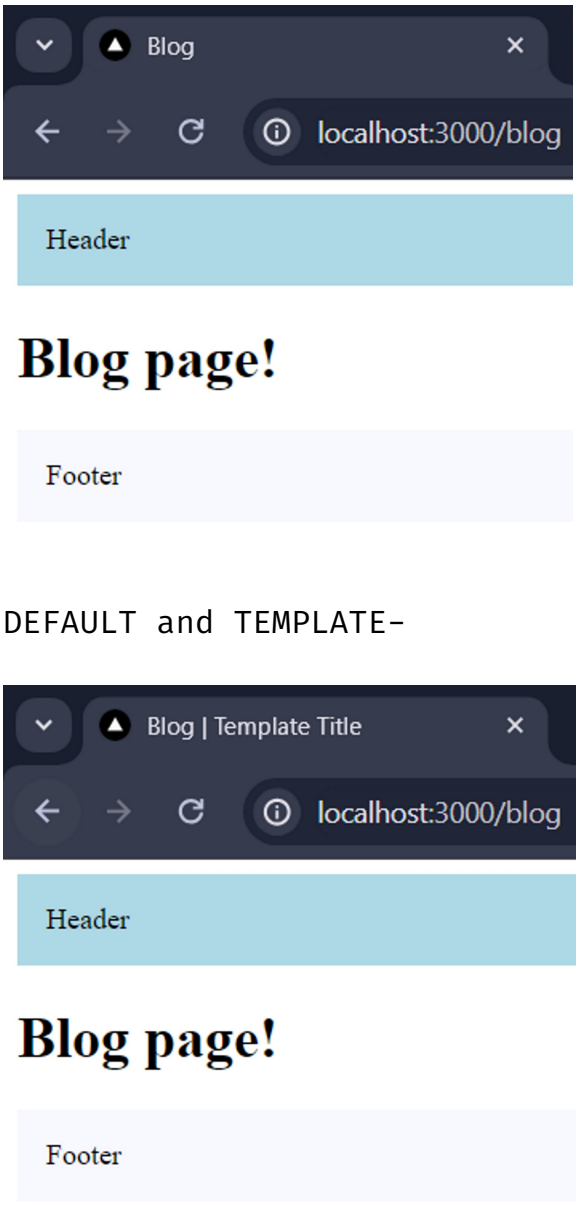
ABOLSUTE-

app > blog > TS page.tsx > Blog

```tsx
import { Metadata } from "next";

export const metadata:Metadata={
    title:{
        absolute:'Blog'
    }
}

export default function Blog(){
    return <h1>Blog page!</h1>
}
```



DEFAULT and TEMPLATE-

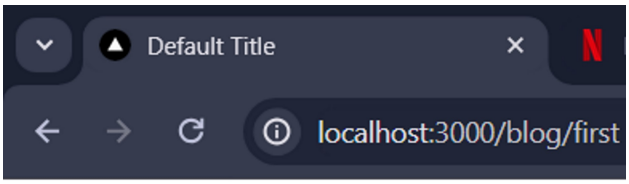
app > layout.tsx > metadata > title

```tsx
import { Inter } from 'next/font/google'
import './globals.css';
import { Metadata } from 'next';

const inter = Inter({ subsets: ['latin'] })

export const metadata :Metadata= {
  title:{
    default:'Default Title',
    template:'%s | Template Title'
  },
  description: 'Generated by create next app',
}

export default function RootLayout({ children }) {
  return (
    <html lang="en">
      <body>
        <header style={{backgroundColor:'lightblue',padding:'1rem'}}>Header</header>
        {children}
        <footer style={{backgroundColor:'ghostwhite',padding:'1rem'}}>Footer</footer>
      </body>
    </html>
  )
}
```



Header

This is first page!

Footer

Component Navigation

Tuesday, December 26, 2023 7:55 PM

We need to navigate through pages through the help of links. In this case the link tag provided by next comes into play. It's like anchor tag used in HTML with a href attribute for the target page.

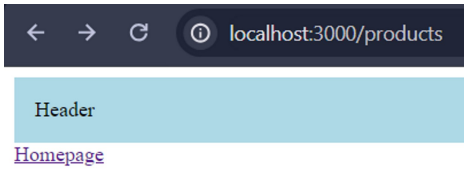
Other action can be to navigate through pages using programmatic navigation after some action is completed.

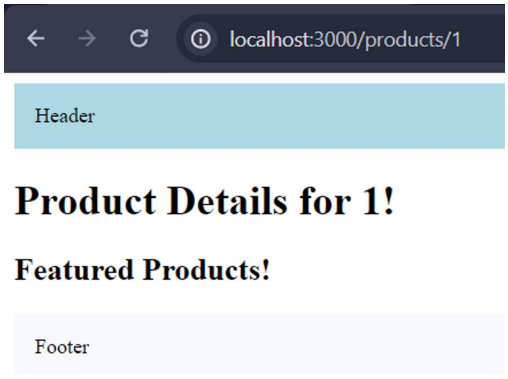
Link navigation-

app > products > page.tsx > Products

```tsx
import Link from "next/link"

export default function Products(){
    return(
        <div>
            <Link href='/'>Homepage</Link>
            <h1>Product List</h1>
            <h2><Link href='products/1'>Product 1</Link></h2>
            <h2><Link href='products/2'>Product 2</Link></h2>
            <h2><Link href='products/3' replace>Product 3</Link></h2>
        </div>
    )
}
```





We include link tag along with the destination path and on click of that it will redirect to the new page.

There is also a replace attribute . If we assign it to a link then on clicking of back button it directly redirects to the homepage

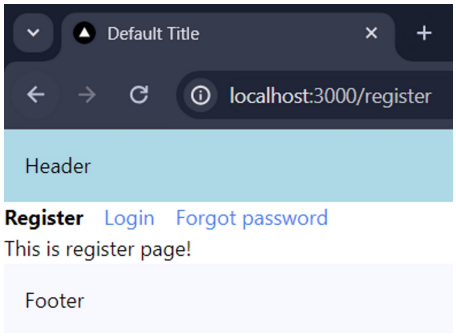
We can check which page of the routing is currently active using pathname imported from next. Based on its value and conditions we can give stylings according to our requirements.

app > (auth) > (with-auth-layout) > layout.tsx > AuthLayout

```tsx
"use client";
import Link from "next/link";
import { usePathname } from "next/navigation";
import "./styles.css";

const navLinks=[
    {name:'Register', href:'/register'},{name:'Login', href:'/login'},{name:'Forgot password', href:'/forgot-password'}
]

export default function AuthLayout({
    children,
}:{
    children:React.ReactNode;
}){
    const pathname=usePathname();
    return(
        <div>
            {navLinks.map((link)=>{
                const isActive=pathname.startsWith(link.href);
                return(
                    <Link href={link.href} key={link.name} className={isActive ? "font-bold mr-4":"text-blue-500 mr-4"}>{link.name}</Link>
                )
            })}
            {children}
        </div>
    )
}
```



Navigating Programmatically-

This case comes into light if we have an event which should be triggered when an action is completed. For eg- redirecting to a new page after a form is submitted.

app > order-product > page.tsx > OrderProduct

```tsx
'use client';

import { useRouter } from "next/navigation";

export default function OrderProduct(){
    const router=useRouter();
    const handleClick=()=>{
        alert('Placing your order!');
        router.push('/');
    }
    return (

        <>
            <h1 className="mt-2 ms-2">Order product</h1>
            <button onClick={handleClick} className="bg-black text-white p-2 mt-3 ms-2">Place order</button>
        </>
    )
}
```

On click of the button, there will be an alert generated and

though the router we move back to the homepage.

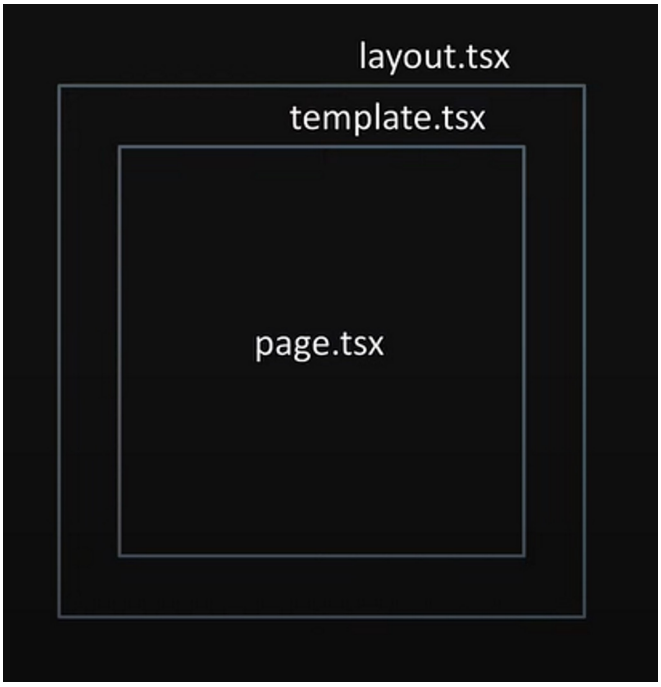
Templates

Wednesday, December 27, 2023 12:29 PM

Templates are similar to layouts in that they wrap each child layout or page
But with templates, when a user navigates between routes that share a template , a new instance of the component is mounted, DOM elements are re-created , state is not preserved and the effects are re-synchronized.

A template can be defined by exporting a default React component from a template.js or template.tsx file.

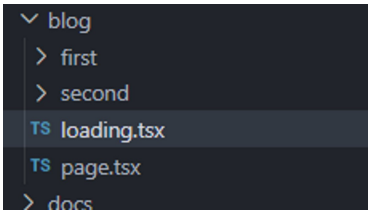
Similar to layouts, templates should accept children prop so that they can render the nested segments in the route.



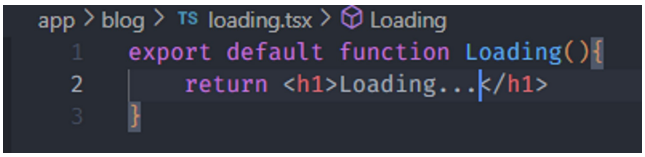
Loading UI

Wednesday, December 27, 2023 1:00 PM

For showing that the application is working and actively loading content, we make a loading.tsx / loading.js file that allows us to create loading states that are displayed to the user while a specific route segment's content is being loaded .



Just place the loading file in the designated folder.



Benefits of using loading-

- You can display loading state as soon as a user navigates to a new route
- Next JS allow the creation of nested layouts that remain interactive while new route segments are loading
-

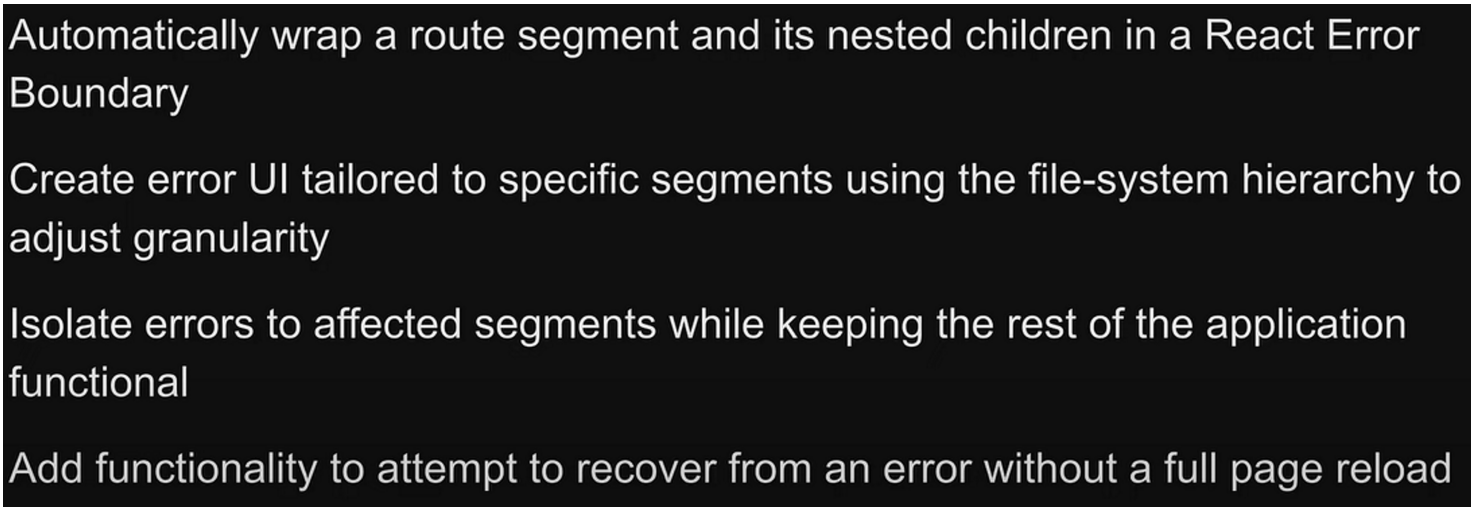
Error Handling

Wednesday, December 27, 2023 1:05 PM

Using a special file called error.tsx / error.js.

Just define it the same way any other file is defined according to the requirements. Remember! Use "use client " at the top for sure.

Work of Error page-



What about handling errors in nested routes?

- Errors pop up to the closest parent boundary
- An error.tsx file will cater to errors for all its nested child elements
- By positioning error.tsx file at different levels in the nested file structure of our project, we can achieve more granular level of error handling

What about handling errors in layouts?

- An error.tsx file will cater to errors for all its nested child elements
- The error boundary does not catch errors here because its nested inside the layouts component

Component Hierarchy (All special files)

Wednesday, December 27, 2023 1:35 PM