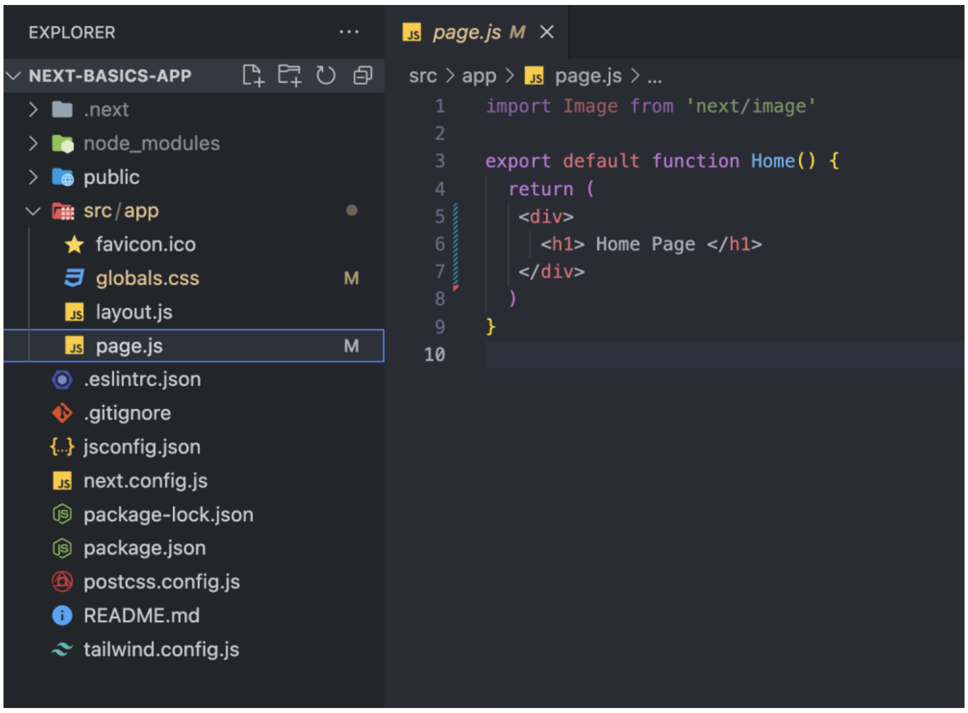
Next Essentials

Folder Structure of a Next App

- Example of folder structure of Next App



```
EXPLORER
NEXT-BASICS-APP
  > .next
  > node_modules
  > public
  v src / app
      favicon.ico
      globals.css        M
      layout.js
      page.js            M
    .eslintrc.json
    .gitignore
    jsconfig.json
    next.config.js
    package-lock.json
    package.json
    postcss.config.js
    README.md
    tailwind.config.js
```

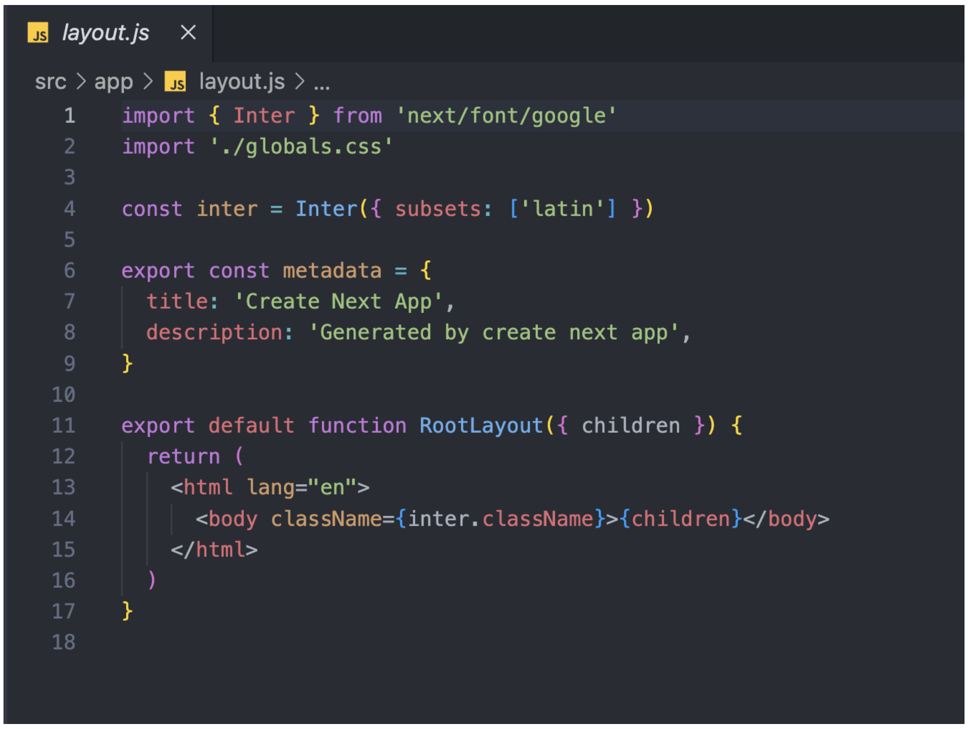
page.js M

src > app > page.js > ...

```
import Image from 'next/image'

export default function Home() {
  return (
   <div>
     <h1> Home Page </h1>
   </div>
  )
}
```

Managing Layouts with Next App

- Example of layouts in Next App



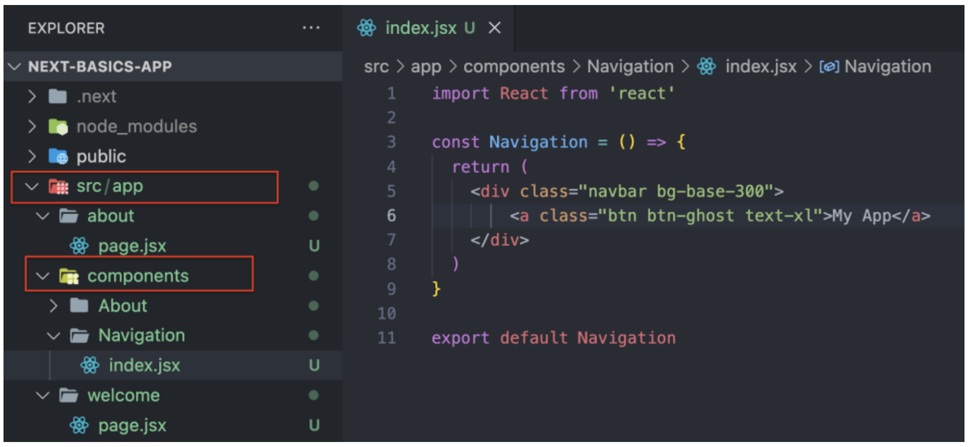
layout.js

src > app > layout.js > ...

```js
import { Inter } from 'next/font/google'
import './globals.css'

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

export const metadata = {
  title: 'Create Next App',
  description: 'Generated by create next app',
}

export default function RootLayout({ children }) {
  return (
    <html lang="en">
      <body className={inter.className}>{children}</body>
    </html>
  )
}

```

Creating Navigation in a Next App

- Example of creating Navigation Component



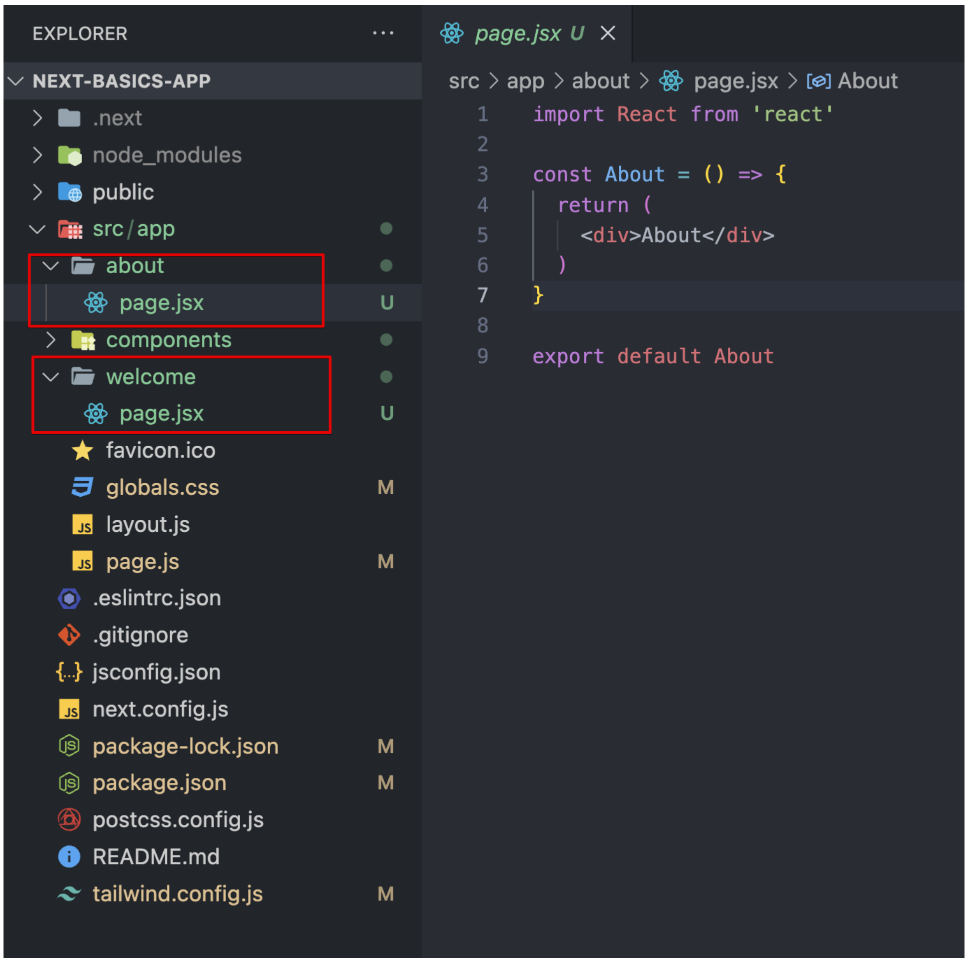
index.jsx U

src > app > components > Navigation > index.jsx > Navigation

```jsx
import React from 'react'

const Navigation = () => {
  return (
    <div class="navbar bg-base-300">
      <a class="btn btn-ghost text-xl">My App</a>
    </div>
  )
}

export default Navigation
```

Creating Pages in a Next App

- In a Next App, version 14 and above. We need to directly create folders for the page with page.jsx as the file inside it. As next uses file-based routing system. It will automatically generate the page after the folder and page.jsx files are created.



```
EXPLORER
NEXT-BASICS-APP
  > .next
  > node_modules
  > public
  v src / app
    v about
        page.jsx                U
    > components
    v welcome
        page.jsx                U
      favicon.ico
      globals.css               M
      layout.js
      page.js                   M
    .eslintrc.json
    .gitignore
    jsconfig.json
    next.config.js
    package-lock.json           M
    package.json                M
    postcss.config.js
    README.md
    tailwind.config.js          M
```

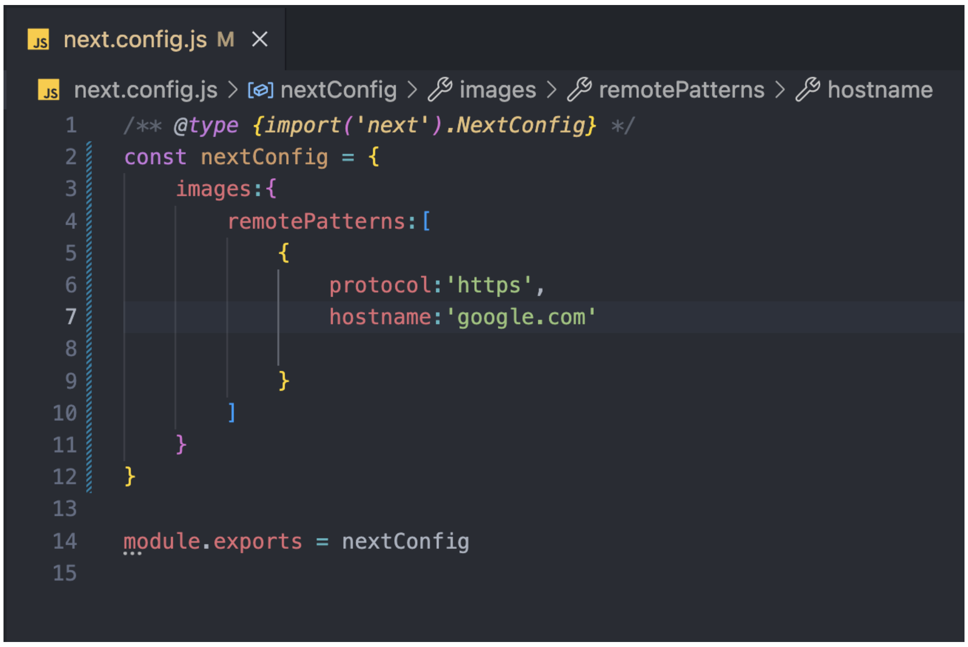
page.jsx U

src > app > about > page.jsx > About

```jsx
import React from 'react'

const About = () => {
  return (
    <div>About</div>
  )
}

export default About
```

Using Next Images for Optimization

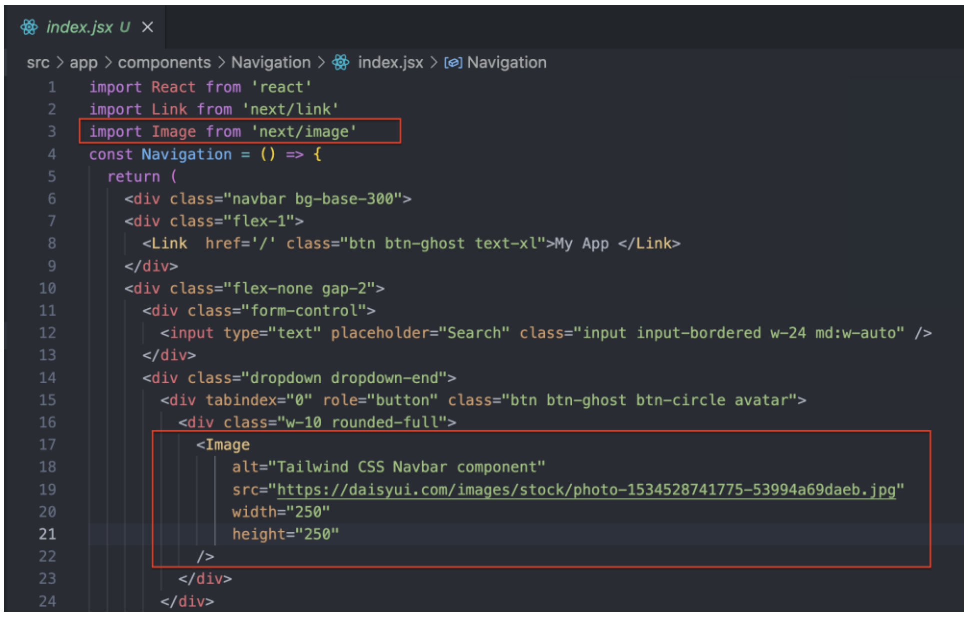
- In a Next App, For navigating between pages we can use the in-built Next Link. Also It is recommended to use Next Images for rendering images in a next.js application.
- Register the domains whose images are going to rendered via CDN in next config file.


next.config.js M
next.config.js > nextConfig > images > remotePatterns > hostname

```js
/** @type {import('next').NextConfig} */
const nextConfig = {
    images:{
        remotePatterns:[
            {
                protocol:'https',
                hostname:'google.com'

            }
        ]
    }
}

module.exports = nextConfig

```

Using Next Images for Optimization

- Next framework provides its own Image component to optimize images in the application to provide high performance and low latency.



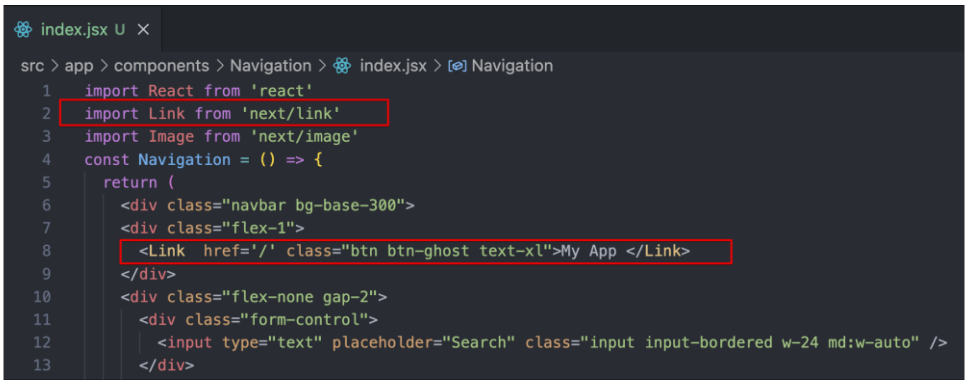
index.jsx U ×

src > app > components > Navigation > index.jsx > Navigation

```jsx
import React from 'react'
import Link from 'next/link'
import Image from 'next/image'
const Navigation = () => {
  return (
  <div class="navbar bg-base-300">
  <div class="flex-1">
    <Link  href='/' class="btn btn-ghost text-xl">My App </Link>
  </div>
  <div class="flex-none gap-2">
    <div class="form-control">
      <input type="text" placeholder="Search" class="input input-bordered w-24 md:w-auto" />
    </div>
    <div class="dropdown dropdown-end">
      <div tabindex="0" role="button" class="btn btn-ghost btn-circle avatar">
        <div class="w-10 rounded-full">
          <Image
            alt="Tailwind CSS Navbar component"
            src="https://daisyui.com/images/stock/photo-1534528741775-53994a69daeb.jpg"
            width="250"
            height="250"
          />
        </div>
      </div>
```

Using Next Link for Navigation

- Example of Next Link for programmatically navigating between pages.



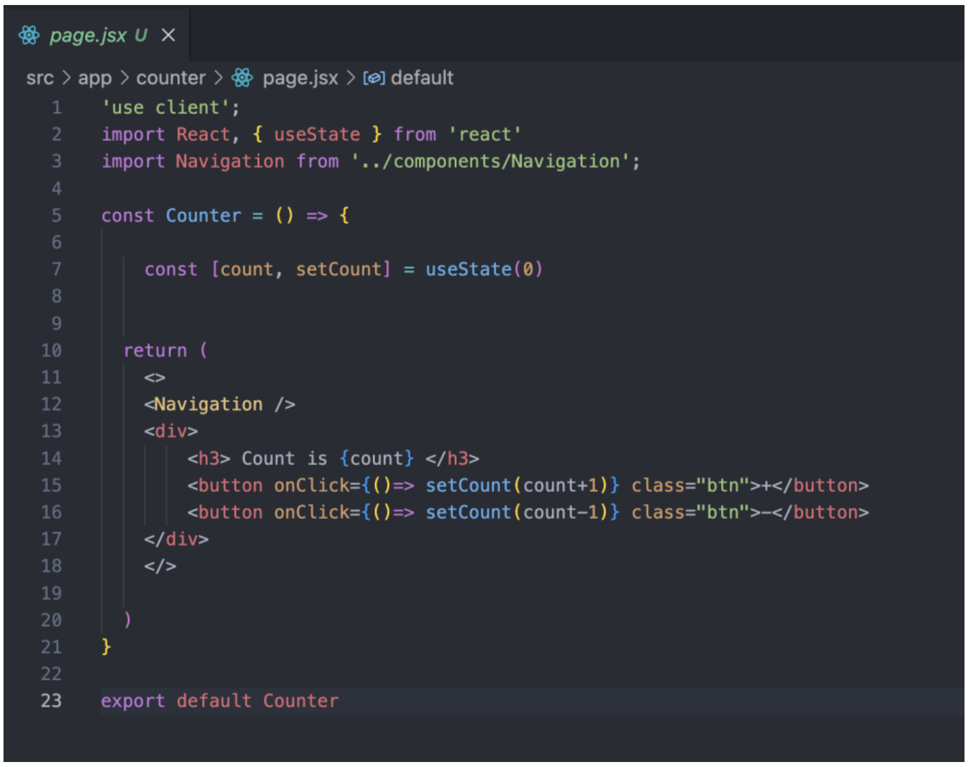
index.jsx U ×

src > app > components > Navigation > index.jsx > Navigation

```jsx
import React from 'react'
import Link from 'next/link'
import Image from 'next/image'
const Navigation = () => {
  return (
  <div class="navbar bg-base-300">
  <div class="flex-1">
    <Link  href='/' class="btn btn-ghost text-xl">My App </Link>
  </div>
  <div class="flex-none gap-2">
    <div class="form-control">
      <input type="text" placeholder="Search" class="input input-bordered w-24 md:w-auto" />
    </div>
```

Create a CSR Component using Next

- In order to create a client side component using next framework. You can add ‘use client’ at the top of the file to inform the next compiler to make the component a client side component.



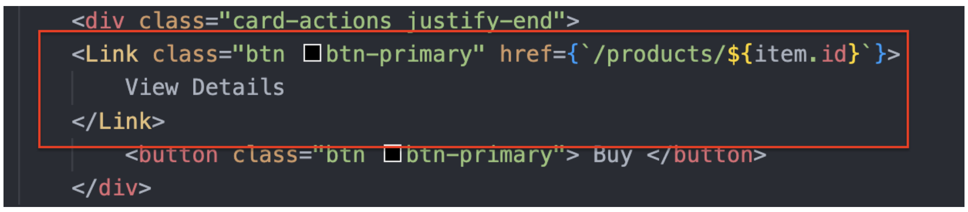
page.jsx U ×

src > app > counter > page.jsx > default

```jsx
'use client';
import React, { useState } from 'react'
import Navigation from '../components/Navigation';

const Counter = () => {

    const [count, setCount] = useState(0)


  return (
    <>
    <Navigation />
    <div>
        <h3> Count is {count} </h3>
        <button onClick={()=> setCount(count+1)} class="btn">+</button>
        <button onClick={()=> setCount(count-1)} class="btn">-</button>
    </div>
    </>

  )
}

export default Counter
```

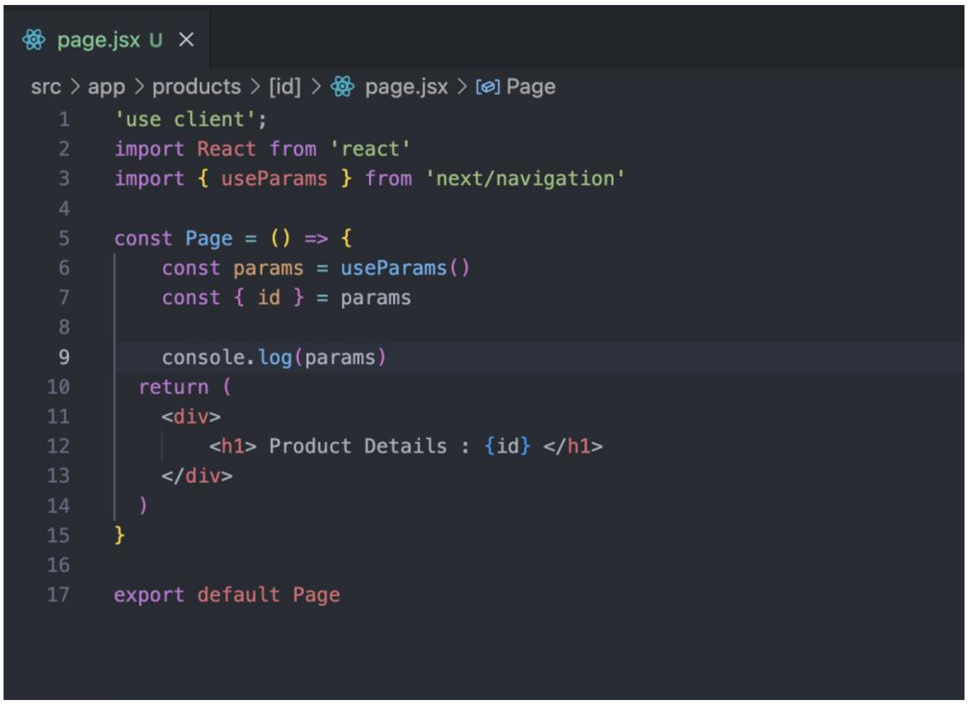
Dynamic Navigation using Next using Route Params

- Example of Dynamic Navigation in Next using Route params



```
<div class="card-actions justify-end">
<Link class="btn btn-primary" href={`/products/${item.id}`}>
    View Details
</Link>
    <button class="btn btn-primary"> Buy </button>
</div>
```

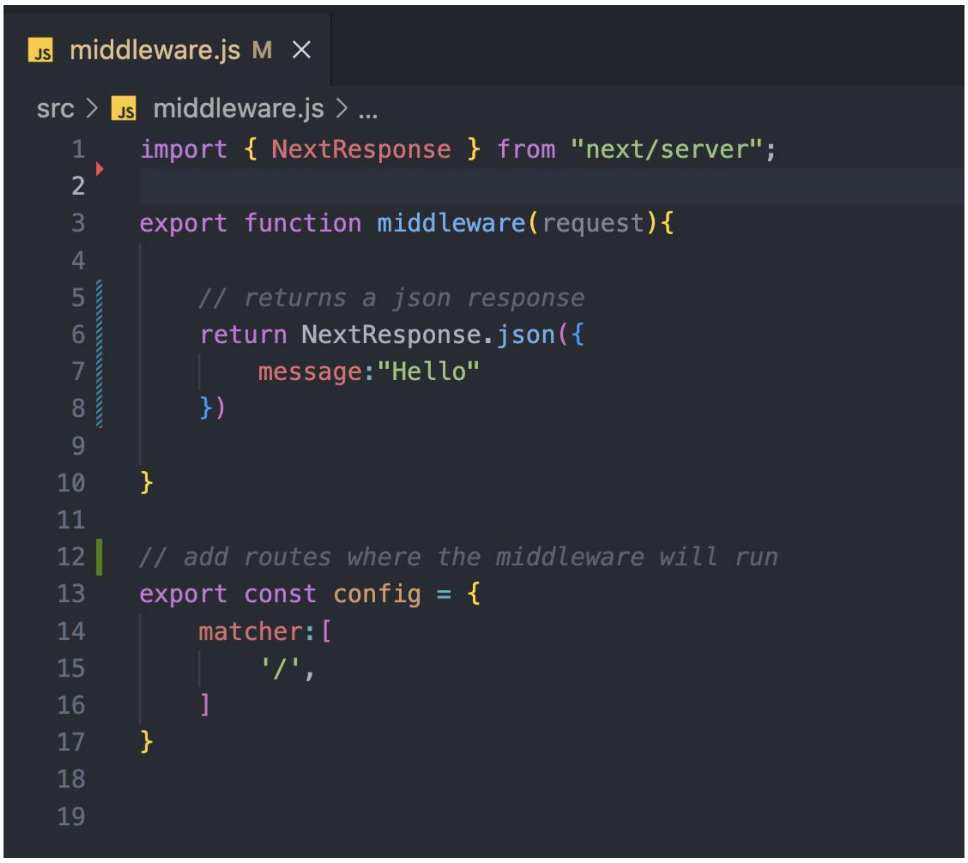
- UseParams() only works with Client Side Components
- For Server Side components you can directly use params as a Prop.



```jsx
'use client';
import React from 'react'
import { useParams } from 'next/navigation'

const Page = () => {
    const params = useParams()
    const { id } = params

    console.log(params)
  return (
    <div>
        <h1> Product Details : {id} </h1>
    </div>
  )
}

export default Page
```

Working with Middlewares in Next App

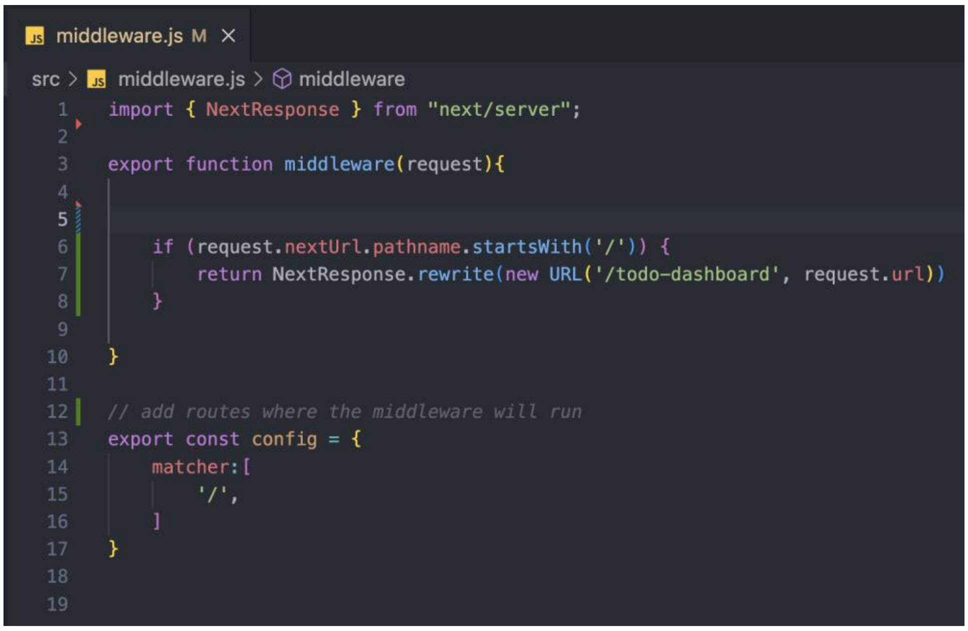
- Example of middleware in Next App



middleware.js M ×

src > middleware.js > ...

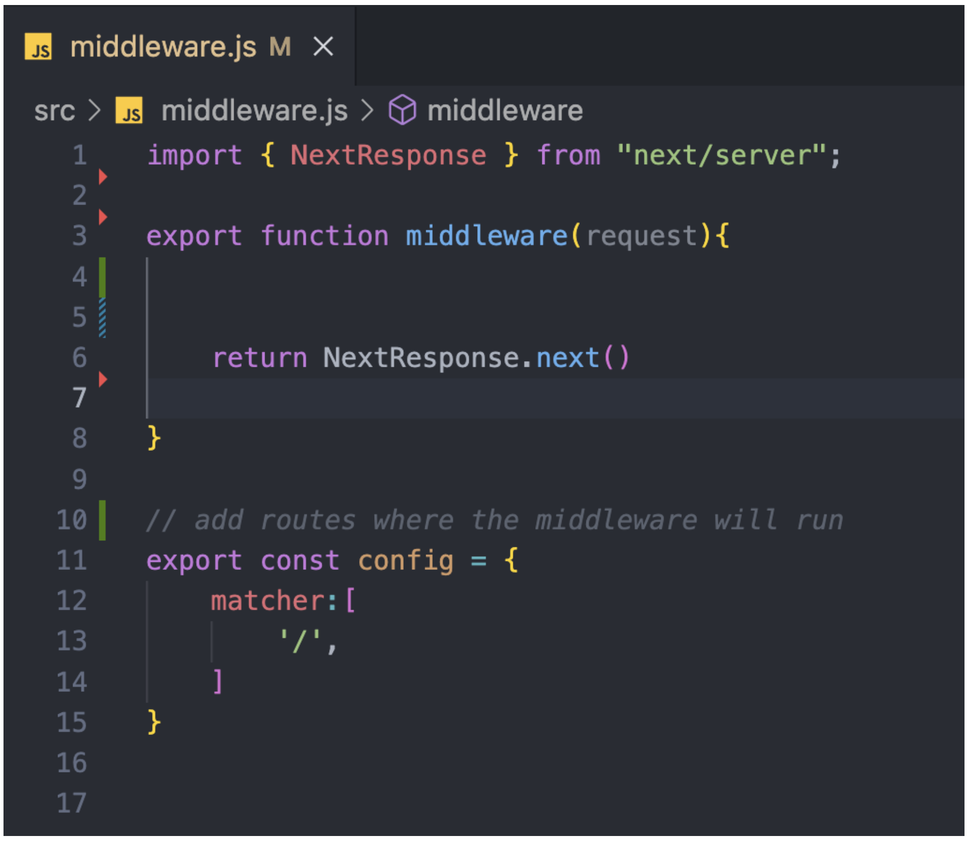
```js
import { NextResponse } from "next/server";

export function middleware(request){

    // returns a json response
    return NextResponse.json({
        message:"Hello"
    })

}

// add routes where the middleware will run
export const config = {
    matcher:[
        '/',
    ]
}
```



middleware.js M ×

src > middleware.js > middleware

```js
import { NextResponse } from "next/server";

export function middleware(request){


    if (request.nextUrl.pathname.startsWith('/')) {
      return NextResponse.rewrite(new URL('/todo-dashboard', request.url))
    }

}

// add routes where the middleware will run
export const config = {
    matcher:[
        '/',
    ]
}
```



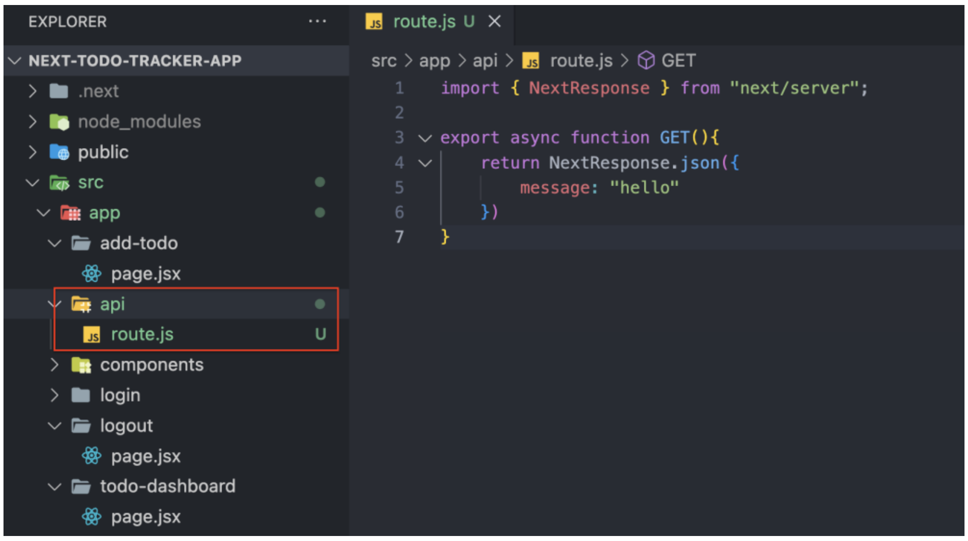
middleware.js M ×

src > middleware.js > middleware

```
import { NextResponse } from "next/server";

export function middleware(request){


    return NextResponse.next()

}

// add routes where the middleware will run
export const config = {
    matcher:[
        '/',
    ]
}
```

Working with API Routes

- Example of API routes in next.js



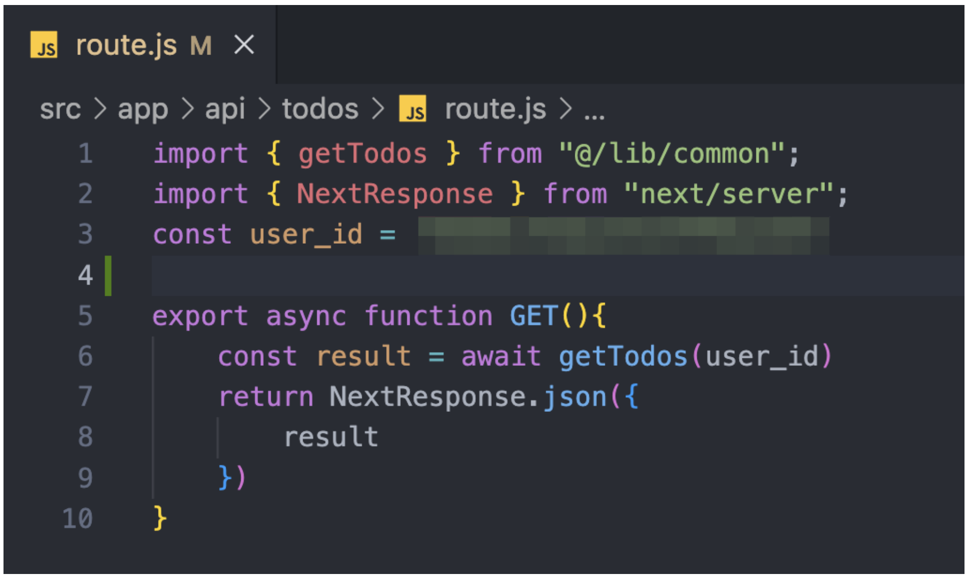
route.js U ×

src > app > api > route.js > GET

```
import { NextResponse } from "next/server";

export async function GET(){
    return NextResponse.json({
        message: "hello"
    })
}
```

GET Request with API Routes using Next 14

- Example



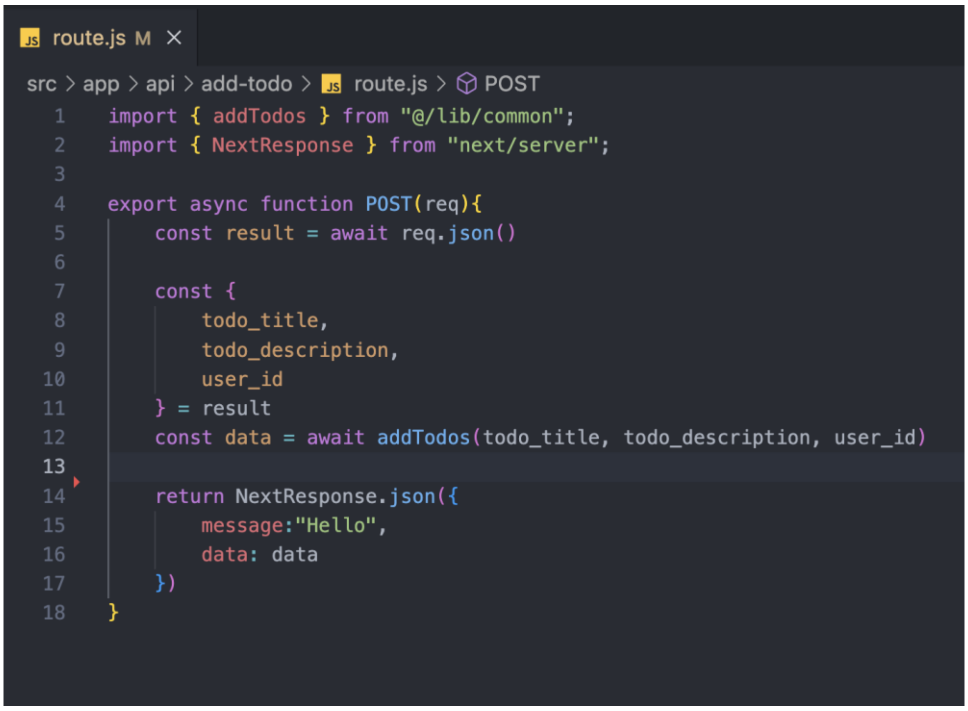
route.js M ×

src > app > api > todos > route.js > ...

```javascript
import { getTodos } from "@/lib/common";
import { NextResponse } from "next/server";
const user_id = [redacted]

export async function GET(){
    const result = await getTodos(user_id)
    return NextResponse.json({
        result
    })
}
```

POST Request with API Routes using Next 14

- Example



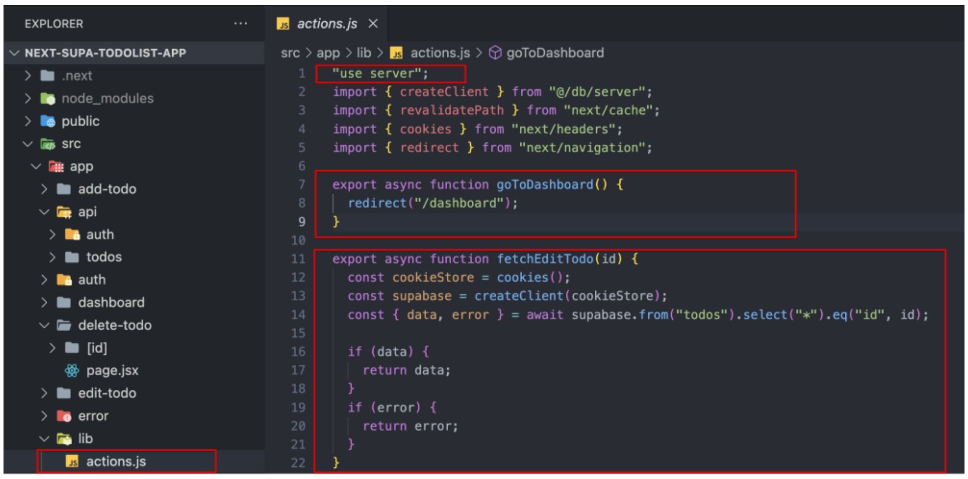
route.js M ×

src > app > api > add-todo > route.js > POST

```javascript
import { addTodos } from "@/lib/common";
import { NextResponse } from "next/server";

export async function POST(req){
    const result = await req.json()

    const {
        todo_title,
        todo_description,
        user_id
    } = result
    const data = await addTodos(todo_title, todo_description, user_id)

    return NextResponse.json({
        message:"Hello",
        data: data
    })
}
```

Server Actions with Next 14

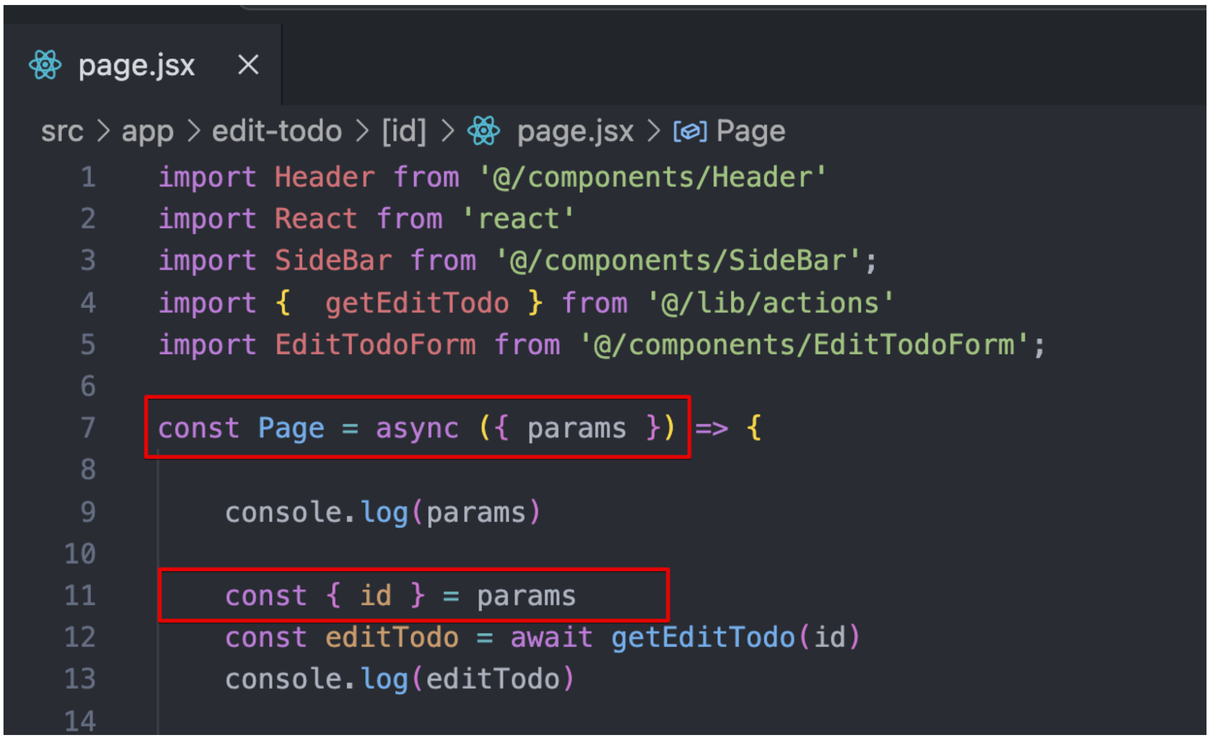
- A server action is a function that will be executed on the server side thus reducing the client side load on the application.
- We can create a server side action by adding ‘use server’ to the top of the function.
- If we have multiple functions, we can include them in a single file and add ‘use server’ on top of the file name.



```javascript
"use server";
import { createClient } from "@/db/server";
import { revalidatePath } from "next/cache";
import { cookies } from "next/headers";
import { redirect } from "next/navigation";

export async function goToDashboard() {
  redirect("/dashboard");
}

export async function fetchEditTodo(id) {
  const cookieStore = cookies();
  const supabase = createClient(cookieStore);
  const { data, error } = await supabase.from("todos").select("*").eq("id", id);

  if (data) {
    return data;
  }
  if (error) {
    return error;
  }
}
```

Access Route Params with Server Components

- In order to access params with Server components, you can directly access the params object when fetching it in the component.
- You can destructure the props of the object and access the Route params.



```jsx
import Header from '@/components/Header'
import React from 'react'
import SideBar from '@/components/SideBar';
import {  getEditTodo } from '@/lib/actions'
import EditTodoForm from '@/components/EditTodoForm';

const Page = async ({ params }) => {

    console.log(params)

    const { id } = params
    const editTodo = await getEditTodo(id)
    console.log(editTodo)

```