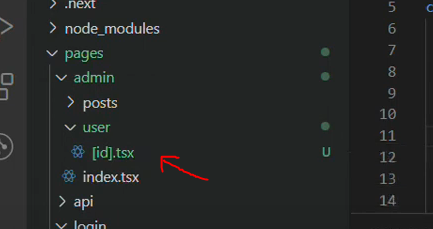
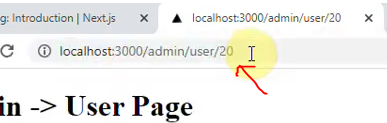
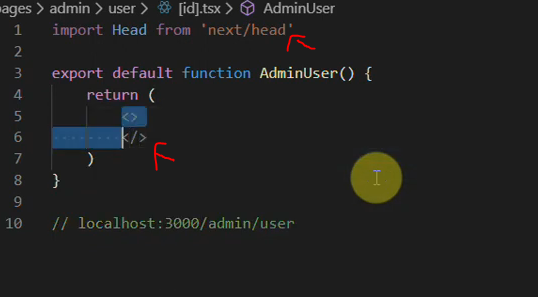


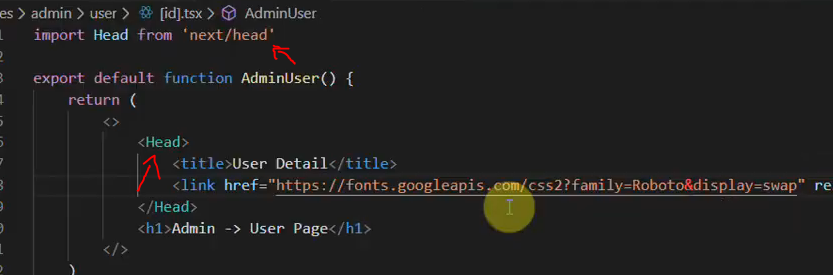
**Tìm hiểu Dynamic Route**





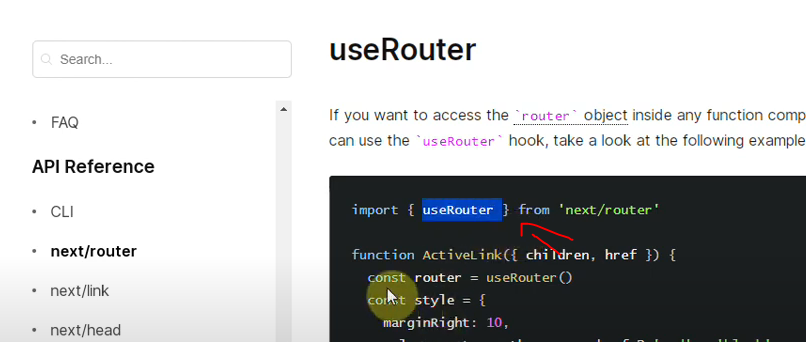


Chỉ dc bọc trpng 1 thẻ

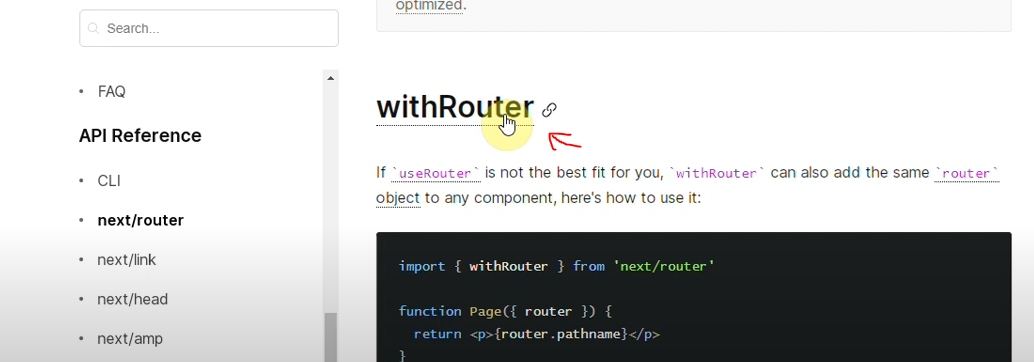


Làm sao để lấy tham số đã truyền :

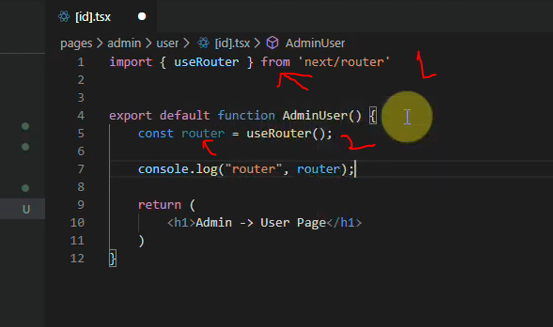
C1 lấy liên quan đến hook



C2 dung cho function arrow

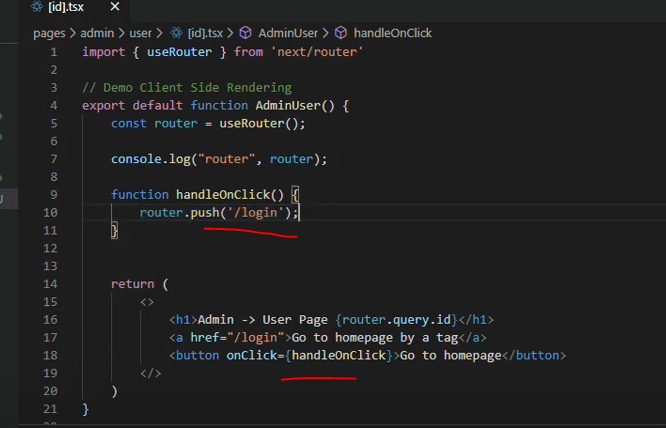


Cách dung

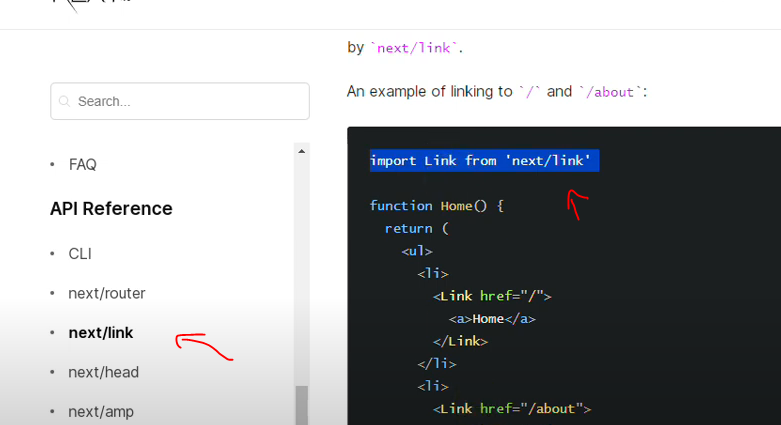


Để chuyển trang

C1



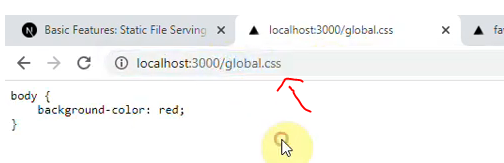
C2



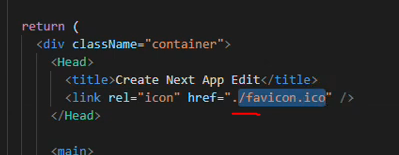
**Tìm hiểu Static File Serving và cách cài đặt: để try cập thông quay link để vào file**

**Để phân biệt đâu là router đau là file**

**Nếu năm trong public thì có thể truy xuất dc**



**Ko ghi ./ nên ghi / thôi => ko dc sửa lại tên**



**State, data của chinh nó,data hiện tại của can ponen chi nó quảng lý, có thể thay dôi**

**Props cũng là trạng thái,nơi khác truyền vào mà do thằng khác truyền vào, ko dc phép thay đổi**

**Compomentwillmount > chạy trước render**

**Compomentdimount> chạy sau ren de**

**Koa**

**npm init -y**

**npm i koa**

**npm i --save-dev nodemon**

**npm i koa-router**

**npm i koa-ejs**

**npm i koa-bodyparser**

**///**

**Chốt**

**yarn create next-app**

npm install next koa koa-router koa-bodyparser

chạy

npm run dev

yarn add react-hook-form

c1 cách làm

Để validate form và lưu và hiển thị dữ liệu trong ứng dụng Next.js và Koa.js, bạn có thể sử dụng các thư viện như Formik và Yup để validate và lưu dữ liệu và sử dụng Koa.js để xử lý các yêu cầu HTTP và truy xuất cơ sở dữ liệu.

Để bắt đầu, hãy làm theo các bước sau:

1. Cài đặt các thư viện:

Bạn cần cài đặt các thư viện sau:

* Formik và Yup để validate và lưu dữ liệu
* Koa.js để xử lý yêu cầu HTTP

Chạy các lệnh sau để cài đặt các thư viện này:

npm install formik yup koa koa-bodyparser koa-router

1. Tạo một trang form

Tạo một trang form với các trường cần thiết. Ví dụ:



import { Formik, Form, Field, ErrorMessage } from 'formik';

import \* as Yup from 'yup';

function MyForm() {

return (

<Formik

initialValues={{ name: '', email: '' }}

validationSchema={Yup.object({

name: Yup.string()

.max(15, 'Must be 15 characters or less')

.required('Required'),

email: Yup.string().email('Invalid email address').required('Required'),

})}

onSubmit={(values, { setSubmitting }) => {

setTimeout(() => {

alert(JSON.stringify(values, null, 2));

setSubmitting(false);

}, 400);

}}

>

{({ isSubmitting }) => (

<Form>

<label htmlFor="name">Name</label>

<Field type="text" name="name" />

<ErrorMessage name="name" />

<label htmlFor="email">Email</label>

<Field type="email" name="email" />

<ErrorMessage name="email" />

<button type="submit" disabled={isSubmitting}>

Submit

</button>

</Form>

)}

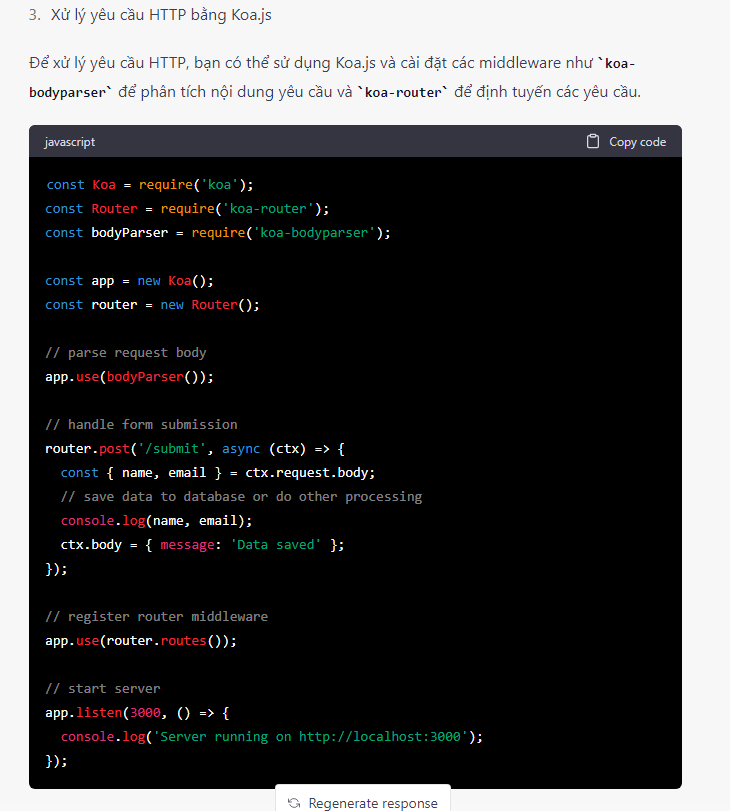
</Formik>

);

}

1. Xử lý yêu cầu HTTP bằng Koa.js

Để xử lý yêu cầu HTTP, bạn có thể sử dụng Koa.js và cài đặt các middleware như **koa-bodyparser** để phân tích nội dung yêu cầu và **koa-router** để định tuyến các yêu cầu.



const Koa = require('koa');

const Router = require('koa-router');

const bodyParser = require('koa-bodyparser');

const app = new Koa();

const router = new Router();

// parse request body

app.use(bodyParser());

// handle form submission

router.post('/submit', async (ctx) => {

const { name, email } = ctx.request.body;

// save data to database or do other processing

console.log(name, email);

ctx.body = { message: 'Data saved' };

});

// register router middleware

app.use(router.routes());

// start server

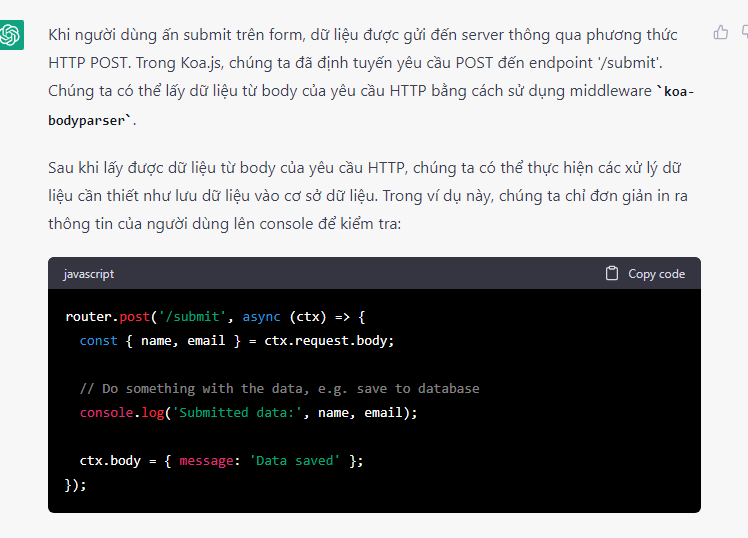
app.listen(3000, () => {

console.log('Server running on http://localhost:3000');

});

Khi người dùng ấn submit trên form, dữ liệu được gửi đến server thông qua phương thức HTTP POST. Trong Koa.js, chúng ta đã định tuyến yêu cầu POST đến endpoint '/submit'. Chúng ta có thể lấy dữ liệu từ body của yêu cầu HTTP bằng cách sử dụng middleware **koa-bodyparser**.

Sau khi lấy được dữ liệu từ body của yêu cầu HTTP, chúng ta có thể thực hiện các xử lý dữ liệu cần thiết như lưu dữ liệu vào cơ sở dữ liệu. Trong ví dụ này, chúng ta chỉ đơn giản in ra thông tin của người dùng lên console để kiểm tra:



router.post('/submit', async (ctx) => {

const { name, email } = ctx.request.body;

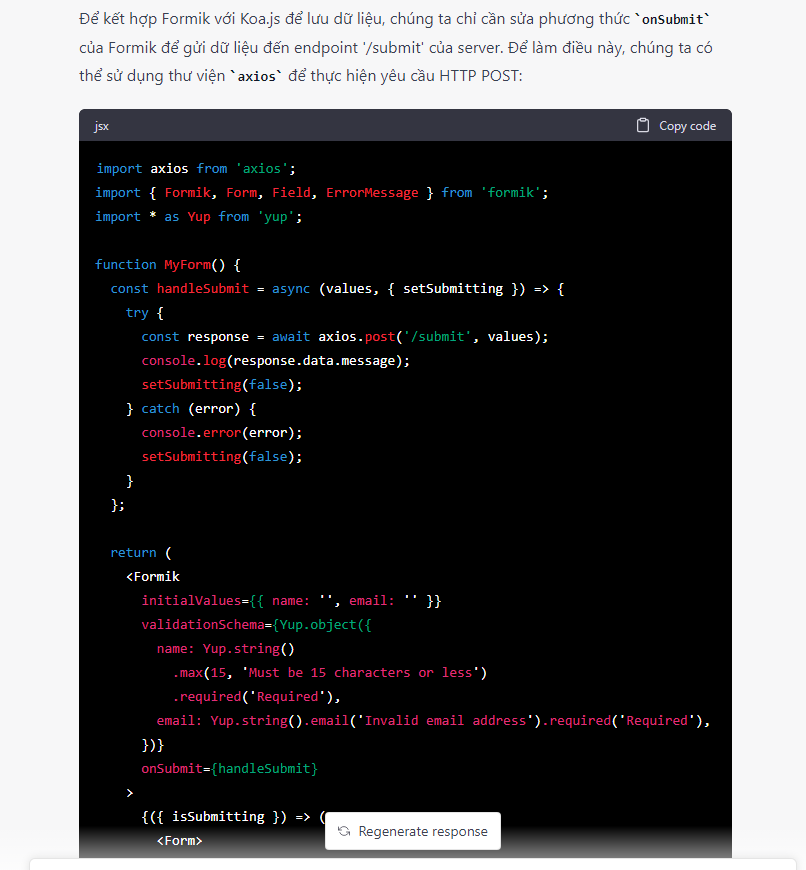
// Do something with the data, e.g. save to database

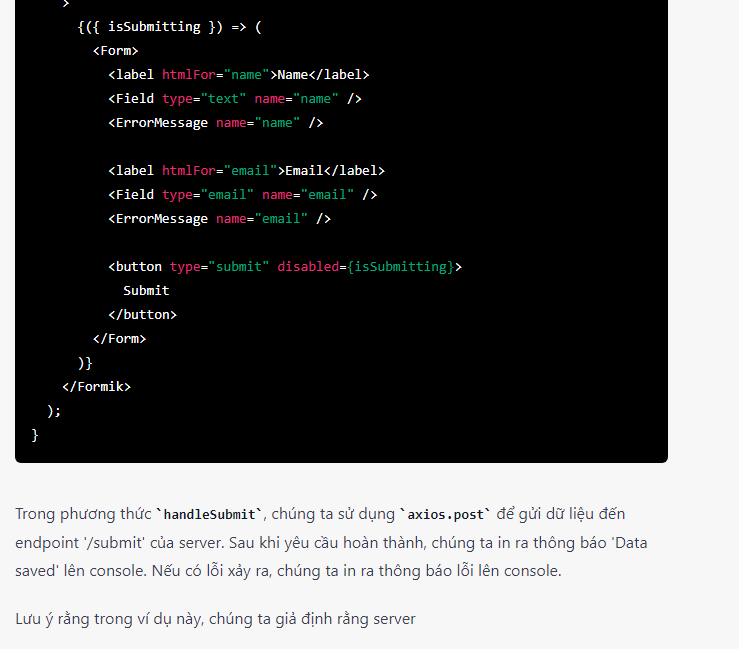
console.log('Submitted data:', name, email);

ctx.body = { message: 'Data saved' };

});

Để kết hợp Formik với Koa.js để lưu dữ liệu, chúng ta chỉ cần sửa phương thức **onSubmit** của Formik để gửi dữ liệu đến endpoint '/submit' của server. Để làm điều này, chúng ta có thể sử dụng thư viện **axios** để thực hiện yêu cầu HTTP POST:





import axios from 'axios';

import { Formik, Form, Field, ErrorMessage } from 'formik';

import \* as Yup from 'yup';

function MyForm() {

const handleSubmit = async (values, { setSubmitting }) => {

try {

const response = await axios.post('/submit', values);

console.log(response.data.message);

setSubmitting(false);

} catch (error) {

console.error(error);

setSubmitting(false);

}

};

return (

<Formik

initialValues={{ name: '', email: '' }}

validationSchema={Yup.object({

name: Yup.string()

.max(15, 'Must be 15 characters or less')

.required('Required'),

email: Yup.string().email('Invalid email address').required('Required'),

})}

onSubmit={handleSubmit}

>

{({ isSubmitting }) => (

<Form>

<label htmlFor="name">Name</label>

<Field type="text" name="name" />

<ErrorMessage name="name" />

<label htmlFor="email">Email</label>

<Field type="email" name="email" />

<ErrorMessage name="email" />

<button type="submit" disabled={isSubmitting}>

Submit

</button>

</Form>

)}

</Formik>

);

}

Trong phương thức **handleSubmit**, chúng ta sử dụng **axios.post** để gửi dữ liệu đến endpoint '/submit' của server. Sau khi yêu cầu hoàn thành, chúng ta in ra thông báo 'Data saved' lên console. Nếu có lỗi xảy ra, chúng ta in ra thông báo lỗi lên console.