Setup a contact form in React that sends email via Node.js

Last modified December 17th 2020 | #node #react | Source Code [GitHub]

Contact forms provide a convenient way for users to get in touch with a website owner. In this tutorial we'll be setting up a simple contact form in a React application. When the form is submitted we'll be sending an email containing the message using Node.js.

Setup the React application

First let's setup the React application using Create React App:

```
npx create-react-app react-contact-form
```

Install dependencies

```
cd react-contact-form
npm install express cors nodemailer
```

- `express` – handles the route used by the POST request.
- `cors` – allows for cross origin resource sharing between the frontend and the server.
- `nodemailer` – simplifies sending emails with Node.js using SMTP.

Create the contact form component

Create a new file called `ContactForm.js` and add the following code:

```
import React, { useState } from "react";

const ContactForm = () => {
  const [status, setStatus] = useState("Submit");
  const handleSubmit = async (e) => {
    e.preventDefault();
    setStatus("Sending...");
    const { name, email, message } = e.target.elements;
    let details = {
      name: name.value,
      email: email.value,
      message: message.value,
    };
    let response = await fetch("http://localhost:5000/contact", {
      method: "POST",
      headers: {
        "Content-Type": "application/json;charset=utf-8",
      },
      body: JSON.stringify(details),
    });
    setStatus("Submit");
    let result = await response.json();
    alert(result.status);
  };
  return (
```

```
      <form onSubmit={handleSubmit}>
        <div>
          <label htmlFor="name">Name:</label>
          <input type="text" id="name" required />
        </div>
        <div>
          <label htmlFor="email">Email:</label>
          <input type="email" id="email" required />
        </div>
        <div>
          <label htmlFor="message">Message:</label>
          <textarea id="message" required />
        </div>
        <button type="submit">{status}</button>
      </form>
    );
  };

  export default ContactForm;
```

This renders a form with three fields (name, email, and message). When the form is submitted we `POST` the values entered into these fields to the Node.js server that we'll setup later on in the tutorial.

Load the component into the application by replacing the contents of `App.js` as follows:

```
  import './App.css';
  import ContactForm from './ContactForm';

  const App = () => {
    return (
      <div className="App">
          <ContactForm />
      </div>
    );
  }

  export default App;
```

Run the `npm start` command to test that the component loads correctly.

With the form created we can move onto processing the submission.

Setup the Node.js server

Create a new file called `server.js` and first load the required dependencies:

```
const express = require("express");
const router = express.Router();
const cors = require("cors");
const nodemailer = require("nodemailer");
```

Next we use `express()` to setup the server that'll run on port 5000:

```
const app = express();
app.use(cors());
app.use(express.json());
app.use("/", router);
app.listen(5000, () => console.log("Server Running"));
```

We'll use Gmail but if you would prefer to use another SMTP service I'd recommend reading the official Nodemailer documentation. You just need to add you username and password here to setup Nodemailer with Gmail:

```
const contactEmail = nodemailer.createTransport({
  service: 'gmail',
  auth: {
    user: "***************@gmail.com",
    pass: "********",
  },
});
```

```
contactEmail.verify((error) => {
  if (error) {
    console.log(error);
  } else {
    console.log("Ready to Send");
  }
});
```

You can now test the server and SMTP connection by running `node server.js`. If successful you'll see "Server Running" & "Ready to Send" logged in the terminal. **NOTE**: If the authentication isn't working you may need to enable less secure apps in your Gmail account here.

To complete the functionality we just need to setup the router and send the email:

```
router.post("/contact", (req, res) => {
  const name = req.body.name;
  const email = req.body.email;
  const message = req.body.message;
  const mail = {
    from: name,
    to: "***************@gmail.com",
    subject: "Contact Form Submission",
    html: `<p>Name: ${name}</p>
           <p>Email: ${email}</p>
           <p>Message: ${message}</p>`,
  };
  contactEmail.sendMail(mail, (error) => {
    if (error) {
      res.json({ status: "ERROR" });
    } else {
      res.json({ status: "Message Sent" });
    }
  });
});
```

If everything was setup correctly when you submit the form you'll get a "Message Sent" alert message and receive an email with the contact form submission.

You now know how to create your own contact form for use in your React applications. If you enjoyed this tutorial there are plenty more tutorials about creating custom React components that can be found here.

Related Posts

- How to create & publish a npm package
- Build a REST API with Node.js, Express, and MongoDB
- How to read and write JSON files using Node.js