Gaurav Singhal

Formspree AJAX with ES6

Gaurav Singhal

Jul 22, 2019 • 8 Min read • 853 Views

Jul 22, 2019 • 8 Min read • 853 Views

Web Development

Introduction

1

Introduction

In this guide, we are going to look into Formspree - a service that grants us simple form submission handling. We will see how we can submit the form to Formspree using ES6 syntax in React.

Formspree

When you are building static sites, you will come across numerous challenges surrounding how to deal with dynamic content, or form submissions. Well, perhaps you'd suggest for contact form submission that we can have a "mailto" link in the form action. We can, but that won't be an excellent experience for the user. The users will be required to switch into their email client and then they will also have to hit the send button- that's too much work to contact someone.

Formspree is the best service for static sites that don't have a backend to handle the form submissions. It is useful for portfolio sites and startups that don't want to invest much. Unlike other alternatives, like Google Form, we don't need to sacrifice the look of our website by embedding forms from external sources using an iframe that doesn't fit in with our User Interface (UI) for contact forms. We can make the form on our own that fits well with our site.

Formspree only requires us to have an action attribute that points to their URL. The URL format is `https://formspree.io/{your email}` and all the forms will be submitted to your email. When the form is submitted for the first time, Formspree will send a confirmation email. Make sure that you confirm by clicking the link in the email to receive form submissions from your website.

Let's Build the Contact Form

jsx

```
<form action="https://formspree.io/your-email@domain.com" method="POST">
            <div className="input-group">
            <label htmlFor="email">E-mail Address</label>
            <input
                type="email"
                name="email"
                id="email"
                value={this.state.values.email}
                onChange={this.handleInputChange}
                title="Email"
                required
            />
            </div>
            <div className="input-group">
            <label htmlFor="message">Message</label>
            <textarea
                name="message"
                id="message"
                onChange={this.handleInputChange}
                title="password"
                required
            >{this.state.values.message}</textarea>
            </div>
            <button type="submit">Send</button>
        </form>
```

For sending the form data using AJAX, we will have to add the submit handler and remove the action attribute.

jsx

```jsx
<form onSubmit={this.submitForm} >
            ...
            <button type="submit">Send</button>
        </form>
```

Like in my previous guide, Submit Form in React without jQuery AJAX we will have the state and handlers as follows:

jsx

```jsx
    this.state = {
        values: {
            email: "",
            message: ""
        },
        isSubmitting: false,
        isError: false
    };
```

jsx

```jsx
    handleInputChange = e =>
        this.setState({
          values: { ...this.state.values, [e.target.name]: e.target.value }
        });
```

```
    submitForm = e => {
        e.preventDefault();
        // fetch() - post data to formspree url
    }
```

In my previous guide, we learned how we could submit form data using the fetch() API. For those of you who don't know what that is, it's the new Promise-based native method to make network requests.

javascript

```
fetch(url, {...options})
    .then(response => {
        // resolve the data
    })
    .then(data => {
        // use the data
    });
```

For this guide, we will be using an external library - **axios** to make the network request for us.

Axios

The problem with fetch is that we require two steps to get the data and make use of it. Suppose, if we have to upload files, fetch won't get us the upload progress. So for such scenarios, we can use Axios.

Internally Axios uses the XML HTTP Request (XHR) and abstracts it's the complex syntax for us and provides a much cleaner method.

Basic Usage

GET Request Example

javascript

```
axios.get(url, {...options})
    .then(response => {
        // handle success
    })
    .catch(error => {
        // handle error
    })
```

POST Request Example

javascript

```
axios.post(url, data, {...options})
    .then(response => {
        // handle success
    })
    .catch(error => {
        // handle error
    })
```

We can also use async/await with Axios:

javascript

```
const getData = async () => {
    try {
                const {data} = await axios.get(url);
                return data;
            } catch(err) {
                console.log(err);
            }
        }
```

submitForm() Handler Using Axios

Now that we know about Axios and how to use its syntax, let's request the Formspree URL.

jsx

```
    submitForm = async e => {
        e.preventDefault();
        try {
            const {data} = await axios.post("https://formspree.io/your-email@
        } catch (err) {
            console.log(err);
        }
```

Complete Code

To summarise all the code above, here's the complete source for this guide.

index.js

jsx

```jsx
import React, { Component } from "react";
import ReactDOM from "react-dom";

import axios from "axios";

import "./styles.css";

class ContactForm extends Component {
  constructor(props) {
    super(props);
    this.state = {
      values: {
        email: "",
        message: ""
      },
      isSubmitting: false,
      isError: false
    };
  }

  submitForm = async e => {
```

```
    e.preventDefault();
    console.log(this.state);
    this.setState({ isSubmitting: true });
    try {
        const {data} = await axios.post("https://formspree.io/your-email@
    } catch (err) {
        console.log(err);
    }

    setTimeout(
      () =>
        this.setState({
          isError: false,
          message: "",
          values: { email: "", password: "" }
        }),
      1600
    );
  };

  handleInputChange = e =>
    this.setState({
      values: { ...this.state.values, [e.target.name]: e.target.value }
    });

  render() {
    return (
      <div>
        <form onSubmit={this.submitForm}>
          <div className="input-group">
            <label htmlFor="email">E-mail Address</label>
            <input
              type="email"
              name="email"
              id="email"
```

```
              value={this.state.values.email}
              onChange={this.handleInputChange}
              title="Email"
              required
            />
          </div>
          <div className="input-group">
            <label htmlFor="message">Message</label>
            <textarea
              name="message"
              id="message"
              onChange={this.handleInputChange}
              title="password"
              required
            >{this.state.values.message}</textarea>
          </div>
          <button type="submit">Send</button>
        </form>
        <div className={`message ${this.state.isError && "error"}`}>
          {this.state.isSubmitting ? "Submitting..." : this.state.message
        </div>
      </div>
    );
  }
}

function App() {
  return (
    <div className="App">
      <h1>Contact US</h1>
      <ContactForm />
    </div>
  );
}

```

```
const rootElement = document.getElementById("root");
ReactDOM.render(<App />, rootElement);
```

styles.css

CSS

```
.App {
  font-family: sans-serif;
}
.input-group {
  margin-bottom: 10px;
}
.input-group label {
  display: block;
  margin-bottom: 5px;
}
button {
  border: none;
  padding: 8px 24px;
}
.message {
  margin-top: 20px;
  font-weight: 600;
}
.message.error {
  color: red;
}
```

Limitations

Unfortunately, Formspree doesn't allow AJAX calls to their URL and access to their API for free users. So, those users will have to make do with the action form attribute.

That's it from this guide. Until next time, keep hustling and code like a beast.

References

- Formspree
- Axios