This guide covers how to create a contact form in a Gatsby site, along with an overview of some strategies for handling form data that has been submitted.

Gatsby is built on top of React. So anything that is possible with a React form is possible in Gatsby. Additional details about how to add forms to your Gatsby site can be found in the Adding Forms section.

Creating an Accessible Form

Faulty forms are a common barrier to a website's accessibility, and can be especially problematic if you use a keyboard and screen reader to navigate the web. Forms should be clearly and intuitively organized into groups of related information, and each form field should be identified with a proper label.

More information on creating accessible forms can be found in WebAIM's article on the subject.

Sending Form Data

When you submit a form, the corresponding data is typically sent to a server to be handled in some manner. More in-depth information on sending form data can be found on MDN.

Each method detailed below will start with the following contact form:

```
<form method="post" action="#">
  <label>
    Name
    <input type="text" name="name" id="name" />
  </label>
  <label>
    Email
    <input type="email" name="email" id="email" />
  </label>
  <label>
    Subject
    <input type="text" name="subject" id="subject" />
  </label>
  <label>
    Message
    <textarea name="message" id="message" rows="5" />
  </label>
  <button type="submit">Send</button>
  <input type="reset" value="Clear" />
</form>
```

Form submission options in Gatsby

Getform

Getform is a form backend platform which offers a free-plan for handling form submissions on static sites. Begin by creating a form on your Gatsby site that you can receive submissions from. When creating the form, direct the HTTP POST method to the Getform, by placing the `name` attributes for the fields you want to make visible. (name, email, message etc.)

```
<form method="post" action="https://getform.io/{your-unique-getform-endpoint}">
  ...
  <label>
    Email
    <input type="email" name="email" />
  </label>
  <label>
    Name
    <input type="text" name="name" />
  </label>
  <label>
    Message
    <input type="text" name="message" />
  </label>
  ...
</form>
```

Once you've made the code changes to your form, you can head over to the contact page on your site and start submitting data to the form. The submissions will then be visible on the Getform dashboard. You can add multiple email addresses to receive email notifications for the forms created, as well as manipulate the data you see on Getform using Zapier and Webhooks options that are offered.

You can find more info on the registration process and form setup on the Getform website and find code examples (AJAX, reCAPTCHA etc) on their CodePen.

Netlify

If you're hosting your site with Netlify, you gain access to their excellent form handling feature.

Setting this up only involves adding a few form attributes:

```
- <form method="post" action="#">
+ <form method="post" netlify-honeypot="bot-field" data-netlify="true"
name="contact">
+   <input type="hidden" name="bot-field" />
+   <input type="hidden" name="form-name" value="contact" />
  ...
```

Now, all submissions to your form will appear in the Forms tab of your site dashboard. By adding the form attribute `netlify-honeypot="bot-field"` and a corresponding hidden input, Netlify will know to quietly reject any spam submissions you may receive.

More information on Netlify Forms can be found on their website.

Formspree

Formspree offers a generous free-tier service for handling form submissions on static sites. This makes it a great tool for having form submissions sent directly to an email address of your choosing, with very little setup required.

In order to begin leveraging Formspree's features, you must add a form action directing the HTTP POST method to the Formspree API (substituting your chosen email), as well as changing the `name` attribute of the email input to `name="_replyto"`.

```
<form method="post" action="https://formspree.io/email@domain.tld">
  ...
  <label>
    Email
    <input type="email" name="_replyto" />
  </label>
  ...
</form>
```

Once you've made the changes you can submit your own form for the first time and register using the email Formspree will send you, and all subsequent form submissions will be sent to your email address. You can find more information on the registration process or setup on their website.

All forms set up in this way come with reCAPTCHA by default, but you can also enable Honeypot spam filtering by adding a hidden input element with the `name="_gotcha"` field.

```
<input type="text" name="_gotcha" style="display:none" />
```

Because the input is hidden, Formspree will know that only a bot could have made the submission and it will be silently ignored!

Run your own server

If your form data requires a significant amount of business logic to handle, creating your own service might make the most sense. The most popular solution to this is writing an HTTP server - this can be done in many languages including PHP, Ruby, GoLang, or in our case Node.js with Express.

An initial implementation of a server using express, body-parser, and nodemailer may look like this:

```
const bodyParser = require("body-parser")
const express = require("express")
const nodemailer = require("nodemailer")

const app = express()
app.use(bodyParser.urlencoded())

const contactAddress = "hey@yourwebsite.com"

const mailer = nodemailer.createTransport({
  service: "Gmail",
  auth: {
    user: process.env.production.GMAIL_ADDRESS,
    pass: process.env.production.GMAIL_PASSWORD,
  },
})

app.post("/contact", function (req, res) {
  mailer.sendMail(
    {
      from: req.body.from,
      to: [contactAddress],
```

```
      subject: req.body.subject || "[No subject]",
      html: req.body.message || "[No message]",
    },
    function (err, info) {
      if (err) return res.status(500).send(err)
      res.json({ success: true })
    }
  )
})

app.listen(3000)
```

This initial implementation listens for POST requests to `/contact`, and sends you an email with the submitted form data. You can deploy this server with services such as Vercel.

Once deployed, note the url of the deployment (something like `my-project-abcd123.vercel.app`), and use it as your form action:

```
<form method="post" action="my-project-abcd123.vercel.app/contact">
  ...
</form>
```

Now, all subsequent form submissions will be sent to your email address!

For an in-depth guide on running your own mail server, you can refer to this awesome guide by DataFire.

Other resources

If you have any issues or if you want to learn more about implementing your own contact form in Gatsby, check out this tutorial from Scott Tolinski:

https://youtu.be/hF7xJhzrr9s