Forms are common to capture user information from site visitors. They range in complexity from simple newsletter email capture boxes, to detailed lead capture forms and signup pages requiring visitors to enter their first & last name, email, and several additional self-identification questions.

The humble `<form>` element has evolved a lot since it was created in 1995. By default, a `<form>` stores its input internally and refresh the page when you `submit`. Today, developers generally want to store input in Javascript so you can validate fields and send the data to a third-party system.

Website forms have several "concerns" like data entry storage, validating fields, form submission, form storage, and other event triggers. This documentation will walk through each concern.

Gatsby uses React, so in a couple places we'll be linking to the React docs for code examples. In addition, the React ecosystem also offers some popular, well-maintained libraries like Formik or Final Form which help with more complex forms.

Form concerns

Client-side data entry

When users type into the input, you want to store it in a JavaScript field that will later be available to your form validation and submission logic.

The "Controlled Components" section of the React documentation provides a good code example here.

```
handleChange(event) {
  this.setState({value: event.target.value});
}
...
 render() {
  return (
    ...
    <input type="text" value={this.state.value} onChange={this.handleChange} />
    ...
  )
 }
```

The `handleChange` function takes the value being entered by the user and stores it internally in the component state.

Field validation

Often you'll want to, eg, ensure that the entry in an "Email" field is a valid email address.

If you need to do validation of one or multiple fields, you may want to use a library like Formik, since adding logic and styling around error messages tends to be fairly boilerplate and repetitive.

Form submission

The default behavior of the HTML form element is to send a POST request to the URL and refresh the page. You'll want to prevent the default behavior and send the information to whatever backend you're using.

You do this by passing a function that prevents page refresh as an `onSubmit` prop. Again from the "Controlled Components" section of the React docs:

```
...
handleSubmit(event) {
  event.preventDefault();
}
...
render() {
  return (
    ...
    <form onSubmit={this.handleSubmit}>
    ...
  )
}
...
```

The `handleSubmit` function calls `preventDefault()` on the event which prevents the page from refreshing. After this, you'll want to insert desired behavior, like sending the data to a third-party service.

You may want to make addition changes to visual form display after submit, like disabling the form submit button after submission, showing a spinning wheel, or changing the button color.

If you need to use any sort of authentication token to submit data, you can use Gatsby Functions (Gatsby's implementation of serverless functions) to run this logic while keeping any of your credentials secure.

Triggering additional events

Sometimes you'll want to trigger additional events off of your form submissions that require authenticated calls to a third-party service.

Some examples:

- sending emails from a service like SendGrid
- server-side validation on form submission (eg, is this email already in our system?)

Users tend to handle these situations in two ways: either they send data to a middleware service like Zapier that can trigger multiple events based on data submission. Alternatively, they use Gatsby Functions to run this logic while keeping auth credentials secure.

Embedding third-party forms vs writing your own forms.

If you're using a marketing automation vendor like Hubspot or Marketo, you have a few options: you can either write your own form and send the data to the vendor's endpoint, or you can drop in the vendor's Javascript.

Using the vendor's form library

Vendors tend to have a JavaScript widget for their forms.

We've found these forms tend to be extremely heavy, weighing several hundred KB. This can often be more than the rest of the page combined, and can delay page load by a full second or two on some devices. For pages where conversion is critical, this can be an unacceptable trade-off.

Sending data

Alternately, these services usually have an endpoint to send data to. If you intend to allow marketers the ability to add fields to the form in eg Hubspot, you'll need to make sure that those fields get pulled into your Gatsby form,

which can require some work or using a plugin like gatsby-source-hubspot-forms.

Further Resources

- Walkthrough of creating a form in React