To use Gatsby, you will need a basic understanding of React components.

The official tutorial is a good place to start.

Why React components?

React's component architecture simplifies building large websites by encouraging modularity, reusability, and clear abstractions. React has a large ecosystem of open source components, tutorials, and tooling that can be used seamlessly for building sites with Gatsby. Gatsby is built to behave almost exactly like a normal React application.

The following model shows how data from a source can be queried by GraphQL for use inside components in the process of building a Gatsby site:

Thinking in React is a good resource for learning how to structure applications with React.

How does Gatsby use React Components?

Everything in Gatsby is built using components.

A basic directory structure of a project might look like this:

```
.
├── gatsby-config.js
├── package.json
└── src
    ├── html.js
    ├── pages
    │   ├── index.js
    │   └── posts
    │       ├── 01-01-2017
    │       │   └── index.md
    │       ├── 01-02-2017
    │       │   └── index.md
    │       └── 01-03-2017
    │           └── index.md
    ├── templates
        └── post.js
```

Page components

Components under `src/pages` become pages automatically with paths based on their file name. For example `src/pages/index.js` is mapped to `yoursite.com` and `src/pages/about.js` becomes `yoursite.com/about/`. Every `.js` or `.jsx` file in the pages directory must resolve to either a string or react component, otherwise your build will fail.

Example:

```
import React from "react"

function AboutPage(props) {
  return (
    <div className="about-container">
```

```
      <p>About me.</p>
    </div>
  )
}

export default AboutPage
```

Page template components

You can programmatically create pages using "page template components". All pages are React components but very often these components are just wrappers around data from files or other sources.

`src/templates/post.js` is an example of a page component. It queries GraphQL for markdown data and then renders the page using this data.

See part six of the tutorial for a detailed introduction to programmatically creating pages.

Example:

```
import React from "react"
import { graphql } from "gatsby"

function BlogPostTemplate(props) {
  const post = props.data.markdownRemark
  return (
    <div>
      <h1>{post.frontmatter.title}</h1>
      <div dangerouslySetInnerHTML={{ __html: post.html }} />
    </div>
  )
}

export default BlogPostTemplate

export const pageQuery = graphql`
  query($slug: String!) {
    markdownRemark(fields: { slug: { eq: $slug } }) {
      html
      frontmatter {
        title
      }
    }
  }
`
```

HTML component

`src/html.js` is responsible for everything other than where Gatsby lives in the `<body />`.

In this file, you can modify the `<head>` metadata and general structure of the document and add external links.

Typically you should omit this from your site as the default `html.js` file will suffice. If you need more control over server rendering, then it's valuable to have an `html.js`.

Example:

```
import React from "react"
import favicon from "./favicon.png"

let inlinedStyles = ""
if (process.env.NODE_ENV === "production") {
  try {
    inlinedStyles = require("!raw-loader!../public/styles.css")
  } catch (e) {
    console.log(e)
  }
}

function HTML(props) {
  let css
  if (process.env.NODE_ENV === "production") {
    css = (
      <style
        id="gatsby-inlined-css"
        dangerouslySetInnerHTML={{ __html: inlinedStyles }}
      />
    )
  }
  return (
    <html lang="en">
      <head>
        <meta charSet="utf-8" />
        <meta name="viewport" content="width=device-width, initial-scale=1.0" />
        {props.headComponents}
        <link rel="icon" href={favicon} />
        {css}
      </head>
      <body>
        <div id="___gatsby" dangerouslySetInnerHTML={{ __html: props.body }} />
        {props.postBodyComponents}
      </body>
    </html>
  )
}
```

These are examples of the different ways React components are used in Gatsby sites. To see full working examples, check out the examples directory in the Gatsby repo.

Non-page components

A Non-page component is one that's embedded inside some other component, forming a component hierarchy. An example would be a Header component that's included in multiple page components. Gatsby uses GraphQL to

enable components to declare the data they need. Using the StaticQuery component or useStaticQuery hook, you can colocate a non-page component with its data.