Foundation Gatsby Projects

Create Four Real Production Websites with Gatsby

Nabendu Biswas

Apress®

Foundation Gatsby Projects

Create Four Real Production Websites with Gatsby

Nabendu Biswas

Apress®

Foundation Gatsby Projects: Create Four Real Production Websites with Gatsby

Nabendu Biswas
Bangalore, India

ISBN-13 (pbk): 978-1-4842-6557-4 ISBN-13 (electronic): 978-1-4842-6558-1
https://doi.org/10.1007/978-1-4842-6558-1

Managing Director, Apress Media LLC: Welmoed Spahr
Acquisitions Editor: Louise Corrigan
Development Editor: James Markham
Coordinating Editor: Nancy Chen

Cover designed by eStudioCalamar

Cover image designed by Freepik (www.freepik.com)

Distributed to the book trade worldwide by Springer Science+Business Media New York, 1 New York Plaza, New York, NY 10004. Phone 1-800-SPRINGER, fax (201) 348-4505, e-mail orders-ny@springer-sbm.com, or visit www.springeronline.com. Apress Media, LLC is a California LLC and the sole member (owner) is Springer Science + Business Media Finance Inc (SSBM Finance Inc). SSBM Finance Inc is a **Delaware** corporation.

For information on translations, please e-mail booktranslations@springernature.com; for reprint, paperback, or audio rights, please e-mail bookpermissions@springernature.com.

Apress titles may be purchased in bulk for academic, corporate, or promotional use. eBook versions and licenses are also available for most titles. For more information, reference our Print and eBook Bulk Sales web page at http://www.apress.com/bulk-sales.

Any source code or other supplementary material referenced by the author in this book is available to readers on GitHub via the book's product page, located at www.apress.com/9781484265574. For more detailed information, please visit http://www.apress.com/source-code.

Printed on acid-free paper

To my wife and kid. This book is affectionately dedicated.

Table of Contents

About the Author

Nabendu Biswas is a full-stack JavaScript developer who has been working in the IT industry for the past 15 years.
He has worked for some of the world's top development firms and investment banks. He currently works as an Associate Architect at Innominds. He is also a passionate tech blogger who publishes on `thewebdev.tech` and is an all-round nerd, passionate about everything JavaScript, React, and Gatsby. You can find him on Twitter `@nabendu82`.

About the Technical Reviewer

Alexander Chinedu Nnakwue has a background in Mechanical Engineering from the University of Ibadan, Nigeria and has been a frontend developer for over three years. He has worked on both web and mobile technologies. He also has experience as a technical author, writer, and reviewer. He enjoys programming for the web, and occasionally, you can find him playing soccer. He was born in Benin City and is currently based in Lagos, Nigeria.

Introduction

I have done quite a bit of freelance work in WordPress development since 2011. The three things that I didn't like about WordPress at that time were that little coding knowledge was required, the sites were slow, and they were easily hacked.

The awesome static site generator GatsbyJS solves all of these problems. It is built with React, so you can utilize all your React knowledge. Plus, it uses the in-demand GraphQL, so you will work with that also. The sites are blazing fast and completely secure. You need a bit of ReactJS knowledge to work with Gatsby, but it adds so much to the React ecosystem. It has a large plugin system like WordPress, which adds functionality. It can be used with a wide range of backend systems, like CMS, Firebase, and many more. In this book, we will first create a simple site using only Gatsby. After that, we will use Stackbit to quickly build a Gatsby site. Then we will build a complex site with all features using the Contentful CMS. The last chapter shows you how to build a video chat site, similar to Skype but using the Twilio service.

CHAPTER 1

Creating an Agency Site

In this chapter, we will build a simple demo agency site (known as a *service company* in India). Although I could use one of the many starter kits available at the Gatsby site that offer complete CSS, I've decided to use a starter kit with minimal CSS so you can learn how to write your own. We are going to do the setup first, and then move on to the basic styles. After that, we will create the sections and pages on the site. Finally, we will deploy the site using Netlify.

The Setup

In this project you can use any IDE like VS code. Everything else we will install through `npm`. Let's start with Gatsby.

First install Gatsby globally by running the following command in your terminal.

```
npm install -g gatsby-cli
```

To create a new Gatsby site, run the following command in the terminal. This is the most basic Gatsby starter kit, with minimal Gatsby plugins installed (more on that later).

```
gatsby new agencyDemo https://github.com/gatsbyjs/gatsby-starter-hello-world
```

Now, go to the directory and run `gatsby develop`. The commands are shown in Listing 1-1.

Listing 1-1. The gatsby develop Command

```
cd agencyDemo
gatsby develop
```

The basic site is now up and running,[1] as shown in Figure 1-1.

[1] `http://localhost:8000/`

N. Biswas, *Foundation Gatsby Projects*, https://doi.org/10.1007/978-1-4842-6558-1_1

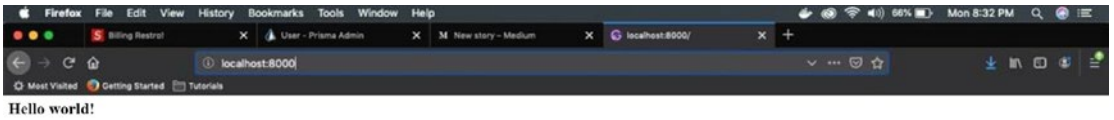

Figure 1-1. *Gatsby is up and running*

We will start by creating the home page.

Creating the Home Page

In Gatsby, everything is React-based, so we will create the home page component. We will first create a set of global styles.

So, first create a folder called `styles` inside `src`. Then create a file called `global.css` inside it. We will put the generic global CSS shown in Listing 1-2 inside this file.

Listing 1-2. The global.css File

```
html {
   box-sizing: border-box;
   font-size: 10px;
   font-weight: 400;
   letter-spacing: 0.075em;
   margin:0;
 }
 *, *:before, *:after {
   box-sizing: inherit;
 }
 body {
   font-family: "Open Sans", Helvetica, sans-serif;
   padding: 0;
   margin: 0;
 }
 a {
   text-decoration: none;
 }
```

Next, create a file called `gatsby-browser.js` in the root directory and include this `global.css` file by adding the following:

```
import "./src/styles/global.css"
```

We will use the extremely popular CSS-in-JS library *styled component* to style the rest of the project.

We need to install some dependencies for `styled-components`. So, open the terminal and type the following command. We have to stop the `gatsby develop` running on the terminal by pressing Ctrl+C.

```
npm install--save gatsby-plugin-styled-components styled-components babel-
plugin-styled-components
```

Next, include the following code in the `gatsby-config.js` file. This file should already be in the root folder.

```
module.exports = {
 plugins: [`gatsby-plugin-styled-components`],
}
```

Then restart `gatsby develop` in the terminal. Next, we will create our index or home page.

Creating the Index Page

We will now start with the index page. We will have a full-page image and some centered text on top of it. First change your `index.js file`, as shown in Listing 1-3.

Listing 1-3. The index.js File

```
import React, { Component } from "react";
import { Link } from "gatsby";
import { Banner, TextWrapper, MoreText } from "../styles/IndexStyles";

export default () => (
   <div style={{position: 'relative'}}>
       <Banner></Banner>
       <TextWrapper>
           <div>
```

```
                <h2>GeekyHacker</h2>
                <p>One Stop for<br/>
                All your development<br />
                And design needs</p>
                <Link to="/works">Our Works</Link>
            </div>
        </TextWrapper>
        <MoreText>Learn More</MoreText>
    </div>
)
```

Now, we will start to write the styled components. Create a file called `IndexStyles.js` inside the `styles` directory.

First we will write styles for the banner. We will show the `banner.jpg` file as a background image, so we will use the `:after` pseudo element.

Also, upload the `banner.jpg` image to the static folder. From the static folder in a Gatsby project, we can directly use an image (see Listing 1-4).

Listing 1-4. The IndexStyles.js File

```
import styled from "styled-components"
const Banner = styled.div`
 &:after {
   content: "";
   display: block;
   height: 100vh;
   width: 100%;
   background-image: url('banner.jpg');
   background-size: cover;
   background-repeat: no-repeat;
   background-position: center;
   filter: grayscale(100%) blur(2px);
 }
```

Next, we will center the text. We need to use the positioning system, as we are showing the text over an image. We already made the parent a `position: relative` and we are making this `div position: absolute`. Then we are using left, top, and transform (see Listing 1-5).

Listing 1-5. The position: absolute Setting

```
const TextWrapper = styled.div`
 position: absolute;
 z-index: 1;
 left: 50%;
 top: 50%;
 transform: translate(-50%, -50%);
 color: white;
 div {
   display: flex;
   justify-content: center;
   align-items: center;
   flex-direction: column;
 }
`;
```

Make sure you have the following at the bottom of the `IndexStyles.js` file or you will get an error.

```
export { Banner, TextWrapper, MoreText }
```

The result is shown in Figure 1-2.

Figure 1-2. *Centered text with the image in the background*

Let's now add some styles to the `h2`, `p`, and `a` tags (`a` is a link tag that converts to an anchor tag). See Listing 1-6.

Listing 1-6. Link to the Anchor Tag

```
h2 {
   font-size: 5rem;
   opacity: 1;
   padding: 0.35em 1em;
   border-top: 2px solid white;
   border-bottom: 2px solid white;
   text-transform: uppercase;
   margin: 0;
 }
 p {
   text-transform: uppercase;
   text-align: center;
   letter-spacing: 0.225em;
```

```
  font-size: 2.5rem;
}
a {
  background-color: #ed4933;
  box-shadow: none;
  color: #ffffff;
  border-radius: 3px;
  border: 0;
  cursor: pointer;
  font-size: 1.5rem;
  font-weight: 600;
  letter-spacing: 0.225em;
  padding: 1.8rem 0.8rem;
  text-align: center;
  text-decoration: none;
  text-transform: uppercase;
}
```

Let's add the style for the `Learn More text`. For this, we also use the `position: absolute` logic, as shown in Listing 1-7.

Listing 1-7. The `Learn More Text`

```
const MoreText = styled.div`
 position: absolute;
 color: #ffffff;
 text-align: center;
 text-transform: uppercase;
 letter-spacing: 0.225em;
 font-weight: 600;
 font-size: 1.2rem;
 z-index: 1;
 left: 50%;
 bottom: 10%;
 transform: translate(-50%, -50%);
```

```
 &:after {
   content: "";
   display: block;
   height: 2rem;
   width: 2rem;
   left: 50%;
   position: absolute;
   margin: 1em 0 0 -0.75em;
   background-image: url("arrow.svg");
   background-size: cover;
   background-repeat: no-repeat;
   background-position: center;
 }
`;

export { Banner, TextWrapper, MoreText };
```

This code will result in the beautiful page shown in Figure 1-3.

Figure 1-3. *The beautiful page*

Creating the Sections

Let's start by creating section two.

Creating Section Two

This section will contain a header, a quotation, and three icons. Update the `index.js` file to include Listing 1-8. The new part is shown in bold.

Listing 1-8. Adding a Header to index.js

```
import React, { Component } from "react";
import { Link } from "gatsby";
import { Banner, TextWrapper, MoreText, SectionTwo } from "../styles/
IndexStyles";

export default () => (
   <>
   <section style={{position: 'relative'}}>
       <Banner></Banner>
       <TextWrapper>
           <div>
               <h2>GeekyHacker</h2>
               <p>One Stop for<br/>
               All your development<br />
               And design needs</p>
               <Link to="/works">Our Works</Link>
           </div>
       </TextWrapper>
       <MoreText>Learn More</MoreText>
   </section>
   <SectionTwo>
       <div>
           <h2>Our Passion</h2>
           <p>Most good programmers do programming not because they expect
           to get paid,
           but because it's fun to program.</p>
```

```
            <h5>- Linus Torvalds</h5>
        </div>
    </SectionTwo>

)
```

Now, let's add some styled components to SectionTwo, as shown in Listing 1-9.

Listing 1-9. Adding Styled Components to index.js

```
const SectionTwo = styled.section`
 background-color: #21b2a6;
 text-align: center;
 padding: 10rem 0;
 div {
   width: 66%;
   margin: 0 auto;
 }
 h2 {
   font-size: 3rem;
   padding: 1.35em 0;
   color: #ffffff;
   border-bottom: 2px solid #1d9c91;
   text-transform: uppercase;
   letter-spacing: 0.6rem;
   margin: 0;
 }
 p {
   text-transform: uppercase;
   color: #c8ece9;
   text-align: center;
   letter-spacing: 0.225em;
   font-size: 1.5rem;
 }
 h5 {
   font-size: 1.4rem;
   line-height: 2rem;
```

```
    color: #ffffff;
    border-bottom: 2px solid #1d9c91;
    font-weight: 800;
    letter-spacing: 0.225em;
    text-transform: uppercase;
    padding-bottom: 0.5rem;
    margin-bottom: 5rem;
  }
`
```

We will now use some `font-awesome` icons to give this section a nice finish. Open the terminal and install these dependencies.

```
npm install @fortawesome/react-fontawesome @fortawesome/fontawesome-svg-core
@fortawesome/free-solid-svg-icons
```

Then import the libraries in `index.js`, as shown in Listing 1-10.

Listing 1-10. Adding Icons to index.js

```
import { FontAwesomeIcon } from '@fortawesome/react-fontawesome';
import { library } from '@fortawesome/fontawesome-svg-core';
import { faHeart, faCode, faGem, fas } from '@fortawesome/free-solid-svg-
icons';

library.add(faHeart, faCode, faGem, fab, fas);
```

Add the code in Listing 1-11, to the `div` and after the `h5`, and you will get the result shown in Figure 1-4.

Listing 1-11. Completing Section Two for index.js

```
<span>
    <FontAwesomeIcon icon="gem" color="#04F5C6" size="6x"
    style={{marginRight: '3rem'}} fixedWidth border />
    <FontAwesomeIcon icon="heart" color="#00F0FF" size="6x"
    style={{marginRight: '3rem'}} fixedWidth border />
    <FontAwesomeIcon icon="code" color="#73DBFD" size="6x"
    fixedWidth border />
</span>
```

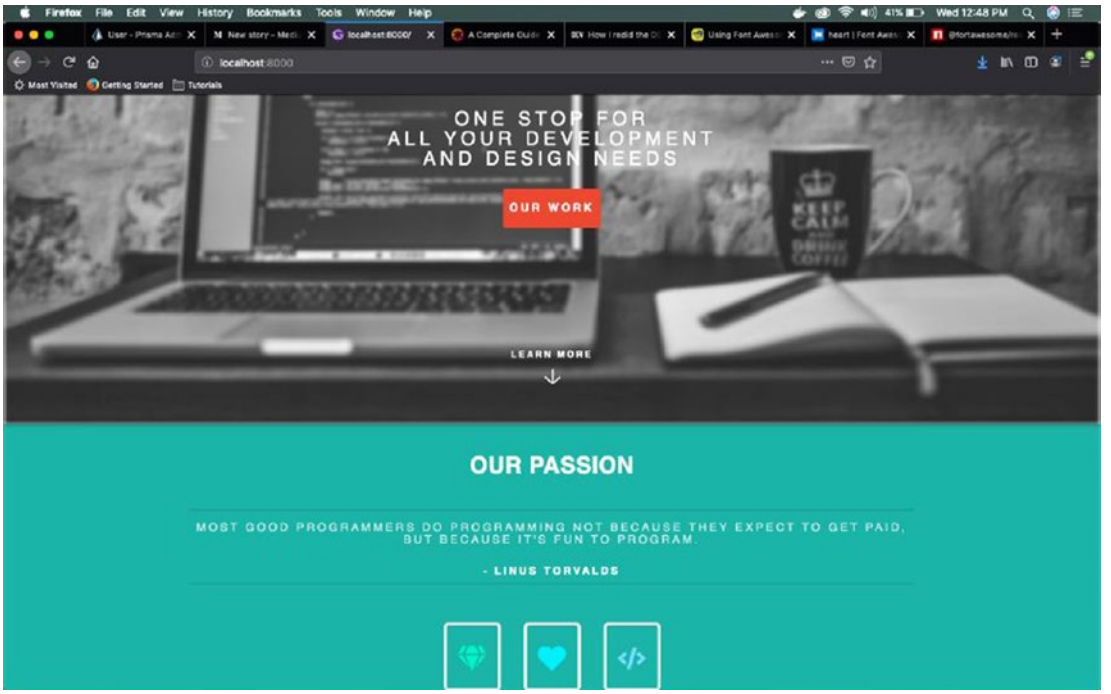

Figure 1-4. *Our passion*

We also need some padding in this section. Let's add it. It is shown in bold in Listing 1-12.

Listing 1-12. Adding Padding to IndexStyles.js

```
const SectionTwo = styled.section`
  background-color: #21b2a6;
  text-align: center;
  padding: 10rem 0;
  div {
    width: 66%;
    margin: 0 auto;
  }
...
...
`
```

Next, we will create Section Three.

Creating Section Three

This section will include an image and text, so we will use Flexbox. Add the code in `index.js`, as shown in Listing 1-13, after `SectionTwo`.

Listing 1-13. Adding Section Three to index.js

```
<SectionThree>
        <FlexBoxIndex>
            <div className="image">
                <img src="pic01.jpg" alt="pic01" />
            </div>
            <div className="text__section3">
                <h2>Website Development</h2>
                <p>
                    We hand code beautiful websites using HTML5,
                    CSS3, JS because they are fully customizable and
                    efficient. No WordPress websites here.
                </p>
            </div>
        </FlexBoxIndex>
        <FlexBoxIndex inverse>
            <div className="text__section3">
                <h2>Website Design</h2>
                <p>
                    We have talented and experienced Web Designers, who
                    can design beautiful and usable websites.
                </p>
            </div>
            <div className="image">
                <img src="pic02.jpg" alt="pic02" />
            </div>
        </FlexBoxIndex>
```

```
            <FlexBoxIndex>
                <div className="image">
                    <img src="pic03.jpg" alt="pic03" />
                </div>
                <div className="text__section3">
                    <h2>Mobile App Development</h2>
                    <p>
                        We develop Mobile apps in Reactive Native, which can
                        be used in both ios and Android.
                    </p>
                </div>
            </FlexBoxIndex>
        </SectionThree>
```

Now, let's add the code in Listing 1-14 to `IndexStyles.js`, just below `SectionTwo`.

Listing 1-14. Adding Section Three to the IndexStyles.js File

```
const SectionThree = styled.section`
 background-color: #2b343d;
 color: #ffffff;
`
const FlexBoxIndex = styled.div`
 display: flex;
 .image {
   width: ${props => (props.inverse ? "60%" : "40%")};
 }
 img {
   width: 100%;
 }
 .text__section3 {
   width: ${props => (props.inverse ? "40%" : "60%")};
   display: flex;
   justify-content: center;
```

```
    align-items: center;
    flex-direction: column;
  }
h2 {
    font-size: 3rem;
    color: #ffffff;
    text-transform: uppercase;
    letter-spacing: 0.225rem;
    margin: 0;
  }
  p {
    text-transform: uppercase;
    color: #c8ece9;
    text-align: center;
    letter-spacing: 0.075em;
    font-size: 1.5rem;
  }
  `
```

Let's do some housekeeping. In Listing 1-15, remove **p** from `TextWrapper`, `SectionTwo`, and `FlexBoxIndex`**.** Then create a new styled component component, called `props`.

Listing 1-15. Housekeeping in IndexStyles.js

```
const GenereicPara = styled.p`
  text-transform: uppercase;
  text-align: center;
  letter-spacing: ${props => (props.lessSpacing ? "0.075em" : "0.225em")};
  font-size: ${props => (props.lessSize ? "1.5rem" : "2.5rem")};
  line-height: ${props => (props.lessSize ? "2rem" : "3rem")};
  color: ${props => (props.grey ? "#c8ece9" : "#ffffff")};
```

Now replace all the `p` tags with `GenericPara`. The updated code is highlighted in bold in Listing 1-16.

Listing 1-16. The index.js File

```
<section style={{ position: 'relative' }}>
    <Banner></Banner>
    <TextWrapper>
        <div>
            <h2>GeekyHacker</h2>
            <GenereicPara>One Stop For<br />
                All your development<br />
                And design needs</GenereicPara>
            <Link to="/works">Our Work</Link>
        </div>
    </TextWrapper>
    <MoreText>Learn More</MoreText>
</section>
<SectionTwo>
    <div>
        <h2>Our Passion</h2>
        <GenereicPara lessSize grey>Most good programmers
        do programming not because they expect to get
        paid,<br />
            but because it's fun to program.</GenereicPara>
        <h5>- Linus Torvalds</h5>
    </div>
    <span>
        <FontAwesomeIcon icon="gem" color="#04F5C6"
        size="6x" style={{marginRight: '3rem'}} fixedWidth
        border />
        <FontAwesomeIcon icon="heart" color="#00F0FF"
        size="6x" style={{marginRight: '3rem'}} fixedWidth
        border />
        <FontAwesomeIcon icon="code" color="#73DBFD"
        size="6x" fixedWidth border />
    </span>
</SectionTwo>
<SectionThree>
```

```
        <FlexBoxIndex>
            <div className="image">
                <img src="pic01.jpg" alt="pic01"/>
            </div>
            <div className="text__section3">
                <h2>Website Development</h2>
                <GenereicPara lessSize lessSpacing>We hand code
                beautiful websites using HTML5, CSS3, JS because
                they are fully customizable and efficient. No
                WordPress websites here.</GenereicPara>
            </div>
        </FlexBoxIndex>
        <FlexBoxIndex inverse>
            <div className="text__section3">
                <h2>Website Design</h2>
                <GenereicPara lessSize lessSpacing>We have
                talented and experienced Web Designers, who
                can design beautiful and usable websites.
                </GenereicPara>
            </div>
            <div className="image">
                <img src="pic02.jpg" alt="pic02"/>
            </div>
        </FlexBoxIndex>
        <FlexBoxIndex>
            <div className="image">
                <img src="pic03.jpg" alt="pic03"/>
            </div>
            <div className="text__section3">
                <h2>Mobile App Development</h2>
                <GenereicPara lessSize lessSpacing>We develop
                Mobile apps in Reactive Native, which can be
                used in both ios and Android.</GenereicPara>
            </div>
        </FlexBoxIndex>
    </SectionThree>
```

Let's do the same song and dance for the h2 tag. Remove h2 from SectionTwo and FlexBoxIndex. Then add a styled component called GenericH2 to the IndexStyles.js file, as shown in Listing 1-17.

Listing 1-17. Adding a Generic h2 Tag

```
const GenericH2 = styled.h2`
 font-size: 3rem;
 padding: ${props => (props.none ? "0" : "1.35em 0")};
 color: #ffffff;
 border-bottom: ${props => (props.none ? "0" : "2px solid #1d9c91")};
 text-transform: uppercase;
 letter-spacing: 0.6rem;
 margin: 0;
`
```

Now replace all the h2 tags with GenericH2. The updated code is highlighted in bold in Listing 1-18.

Listing 1-18. Adding GenericH2

```
<SectionTwo>
    <div>
        <GenericH2>Our Passion</GenericH2>
        <GenereicPara lessSize grey>Most good programmers
        do programming not because they expect to get
        paid,<br />
            but because it's fun to program.</GenereicPara>
        <h5>- Linus Torvalds</h5>
    </div>
  ...
</SectionTwo>
<SectionThree>
    <FlexBoxIndex>
        <div className="image">
            <img src="pic01.jpg" alt="pic01"/>
        </div>
```

```
            <div className="text__section3">
                <GenericH2 none>Website Development</GenericH2>
                <GenereicPara lessSize lessSpacing>We hand
                code beautiful websites using HTML5, CSS3,
                JS because they are fully customizable and
                efficient. No WordPress websites here.
                </GenereicPara>
            </div>
        </FlexBoxIndex>
        <FlexBoxIndex inverse>
            <div className="text__section3">
                <GenericH2 none>Website Design</GenericH2>
                <GenereicPara lessSize lessSpacing>We have
                talented and experienced Web Designers, who
                can design beautiful and usable websites.
                </GenereicPara>
            </div>
            <div className="image">
                <img src="pic02.jpg" alt="pic02"/>
            </div>
        </FlexBoxIndex>
        <FlexBoxIndex>
            <div className="image">
                <img src="pic03.jpg" alt="pic03"/>
            </div>
            <div className="text__section3">
                <GenericH2 none>Mobile App Development
                </GenericH2>
                <GenereicPara lessSize lessSpacing>We develop
                Mobile apps in Reactive Native, which can be
                used in both ios and Android.</GenereicPara>
            </div>
        </FlexBoxIndex>
    </SectionThree>
```

The result is shown in Figure 1-5.