



Advanced Gatsby Projects

Create Two Advanced Sites Using
Technologies that Compliment Gatsby

—

Nabendu Biswas

Apress®

Advanced Gatsby Projects

Create Two Advanced Sites
Using Technologies that
Compliment Gatsby

Nabendu Biswas

Apress®

Advanced Gatsby Projects: Create Two Advanced Sites Using Technologies that Compliment Gatsby

Nabendu Biswas
Bangalore, India

ISBN-13 (pbk): 978-1-4842-6639-7
<https://doi.org/10.1007/978-1-4842-6640-3>

ISBN-13 (electronic): 978-1-4842-6640-3

Copyright © 2021 by Nabendu Biswas

This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed.

Trademarked names, logos, and images may appear in this book. Rather than use a trademark symbol with every occurrence of a trademarked name, logo, or image we use the names, logos, and images only in an editorial fashion and to the benefit of the trademark owner, with no intention of infringement of the trademark.

The use in this publication of trade names, trademarks, service marks, and similar terms, even if they are not identified as such, is not to be taken as an expression of opinion as to whether or not they are subject to proprietary rights.

While the advice and information in this book are believed to be true and accurate at the date of publication, neither the authors nor the editors nor the publisher can accept any legal responsibility for any errors or omissions that may be made. The publisher makes no warranty, express or implied, with respect to the material contained herein.

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/9781484266397. For more detailed information, please visit <http://www.apress.com/source-code>.

Printed on acid-free paper

To my Dad. I miss him every day! He left for heaven in November 2020 due to a stroke. He had retired 10 years back from a top government organization, BHEL, as a senior engineer. He always was the pillar in my life. I hoped to spend more time with him, but life had some other plans.

Table of Contents

About the Author	ix
About the Technical Reviewer	xi
Introduction	xiii
Part I: Creating an Ecommerce Feature Site with Snipcart	1
Chapter 1: Setting up the Ecommerce Site.....	3
Getting Started.....	3
Managing the Default Files	4
Installing react-icons	8
Summary.....	9
Chapter 2: Adding Core Features to the Ecommerce Site.....	11
Navbar.....	11
Basic Setup	11
NavbarHeader Component.....	16
NavbarLinks Component	20
NavbarIcons Component	29
Displaying a Center Image	33
Banner Text and Button.....	41
Banner Component.....	41
Button Component.....	48
Home Component	51
Creating the Footer	62
Deploying in Netlify	66
Gallery Component.....	75
Summary.....	86

TABLE OF CONTENTS

- Chapter 3: Setting up Contentful 87**
 - Creating a Content Model 87
 - Adding Items 104
 - API Keys 112
 - API Keys in Code 115
 - Showing Items 122
 - Summary 131

- Chapter 4: Using Webhooks at the Site 133**
 - Getting Started 133
 - Summary 147

- Chapter 5: Making the Site Dynamic with Snipcart..... 149**
 - Adding a Cart Icon 149
 - Adding an Order Button 154
 - Adding Ecommerce Features 157
 - Additional Settings 168
 - Contact Component 173
 - About Component 176
 - Summary 181

- Part II: Creating a Recipe Website with Firebase 183**
 - Chapter 6: Setting up the Recipe Site 185**
 - Getting Started 185
 - Setting up Firebase 186
 - Completing the Setup 204
 - Summary 211

Chapter 7: Displaying Recipes from Firebase	213
Updating the Code.....	213
Creating Individual Pages	216
Changing the Title and Footer	220
Showing Recipe Details	222
Updating the Home Page	228
Summary.....	233
Chapter 8: Displaying Images from Firebase	235
Storing Images in Firebase	235
Adding the Images in Collections.....	240
Using gatsby-image	253
Showing Images Inside Pages.....	262
Summary.....	266
Chapter 9: Deploying the Recipe Site in Netlify	267
Creating an Environment File.....	267
Deployment Time	270
Summary.....	279
Chapter 10: Adding Disqus Commenting System	281
Modifying the Home Page.....	281
Adding Disqus	285
Summary.....	293
Index	295

About the Author



Nabendu Biswas is a full-stack JavaScript developer who has been working in the information technology industry for the past 15 years. He has worked for some of the world's top development firms and investment banks. He is currently working as an Associate Architect at Innominds. He is a passionate tech blogger who publishes on `thewebdev.tech`. He is a tech YouTuber with a channel named The Web Dev, and also loves to teach people web development. He is an all-around 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 front-end developer for more than three years, working 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 also find him playing soccer. He was born in Benin City and is currently based in Lagos, Nigeria.

Introduction

This book contains two advanced Gatsby projects and it is advisable for beginners to Gatsby to start with my book *Foundation Gatsby Projects*. In the first project, we will create a fully functional e-commerce site for a restaurant using Snipcart. Through this, site a user can place an order and also pay via credit card. It also has a nice dashboard and email notification for the owner of the restaurant.

In the second project, you will learn to build a recipe site using the awesome and free-to-use Firebase real-time database. With Firebase we can use a back end without complicated back ends like Java or NodeJS. We are also going to add a commenting system in this project with Disqus.

PART I

Creating an Ecommerce Feature Site with Snipcart

In Part I, we will build a restaurant site with GatsbyJS. This site will also have an ecommerce feature that allows users to order online. We will be using Snipcart for this feature. We are also going to deploy the project in Netlify and store the data in the Contentful content management system (CMS). We are also going to use webhooks for automatic adding, editing, and deleting of items from the Contentful CMS.

CHAPTER 1

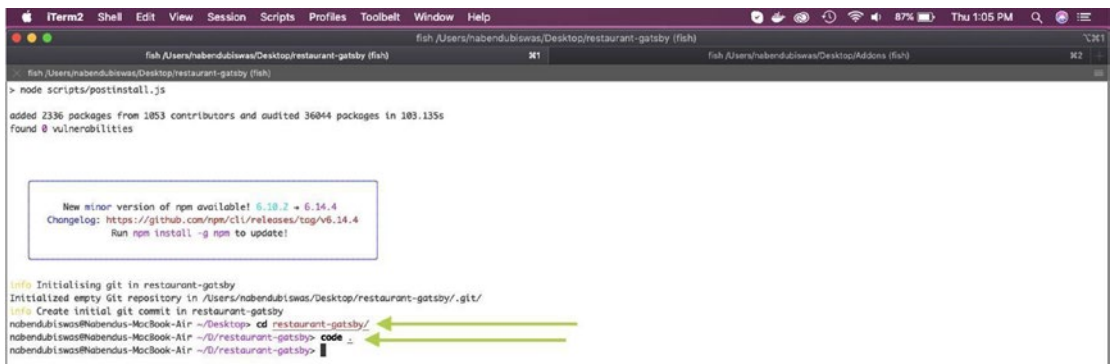
Setting up the Ecommerce Site

In this chapter we will cover how to create a new project, then manage default files, and finally install react-icons into the project.

Getting Started

We first need to create a new project with the familiar `gatsby new <project_name>` command. I used the command in my Desktop with the command `gatsby new restaurant-gatsby`.

Next, let's navigate to the directory and open the project in VS Code (Figure 1-1).



```
fish /Users/nabendubiswas/Desktop/restaurant-gatsby (fish)
fish /Users/nabendubiswas/Desktop/restaurant-gatsby (fish)
> node scripts/postinstall.js
added 2336 packages from 1053 contributors and audited 36044 packages in 103.135s
found 0 vulnerabilities

New minor version of npm available! 5.10.2 > 6.14.4
ChangeLog: https://github.com/npm/cli/releases/tag/v6.14.4
Run npm install -g npm to update!

Info: Initialising git in restaurant-gatsby
Initialized empty Git repository in /Users/nabendubiswas/Desktop/restaurant-gatsby/.git/
Info: Create initial git commit in restaurant-gatsby
nabendubiswas@Nabendus-MacBook-Air ~/Desktop> cd restaurant-gatsby/
nabendubiswas@Nabendus-MacBook-Air ~/D/restaurant-gatsby> code .
nabendubiswas@Nabendus-MacBook-Air ~/D/restaurant-gatsby>
```

Figure 1-1. `cd` and `code`

It is showing perfectly at `http://localhost:8000/` with the starter project (Figure 1-2).

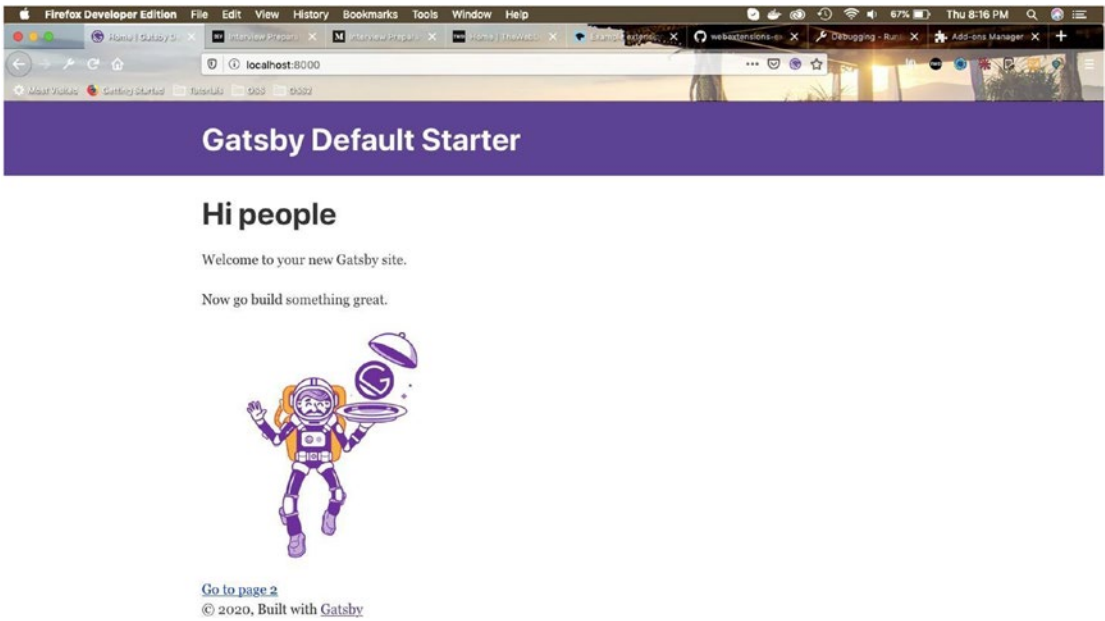


Figure 1-2. Gatsby starter

Managing the Default Files

It's time to update and delete some default files. First open the `gatsby-config.js` file and change the content as shown in bold in Listing 1-1.

Listing 1-1. gatsby-config.js

```
module.exports = {  
  siteMetadata: {  
    title: `Restaurant Site`,  
    description: `The Restaurant Site`,  
    author: `@thewebdev`,  
  },  
  ...  
  ...  
}
```

Next, remove `page-2.js` inside the `pages` folder. Also, update `layout.js` to contain the minimum code for now, as shown in Listing 1-2.

Listing 1-2. layout.js

```
import React from "react"
import PropTypes from "prop-types"
import "../layout.css"

const Layout = ({ children }) => {
  return (
    <main>{children}</main>
  )
}

Layout.propTypes = {
  children: PropTypes.node.isRequired,
}

export default Layout
```

Next, open the `index.js` file and put code shown in Listing 1-3 in it, after removing everything else.

Listing 1-3. index.js

```
import React from "react"
import Layout from "../components/layout"
import SEO from "../components/seo"

const IndexPage = () => (
  <Layout>
    <SEO title="Home" />
    <h3>The Restaurant Site</h3>
  </Layout>
)

export default IndexPage
```

Delete everything inside `layout.css`, and replace it with the content given in Listing 1-4.

Listing 1-4. layout.css

```
* {
  margin: 0;
  padding: 0;
  box-sizing: border-box;
}

body{
  font-family: 'Caveat', cursive;
}
```

Now, we are mainly going to use styled components in our project. As per the [documentation](https://www.gatsbyjs.org/packages/gatsby-plugin-styled-components/?=styled) provided at <https://www.gatsbyjs.org/packages/gatsby-plugin-styled-components/?=styled>, we need to do `npm install` first.

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

So, stop `gatsby develop` and install the plug-in. Now, we have to open `gatsby-config.js` and add the lines shown in bold in Listing 1-5.

Listing 1-5. gatsby-config.js

```
module.exports = {
  siteMetadata: {
    title: `Restaurant Site`,
    description: `The Restaurant Site`,
    author: `@thewebdev`,
  },
  plugins: [
    ...
    ...
    `gatsby-plugin-sharp`,
    {
      resolve: `gatsby-plugin-styled-components`,
      options: {
        // Add any options here
      },
    },
  ],
}
```

```

    },
    {
      resolve: `gatsby-plugin-manifest`,
      options: {
        name: `gatsby-starter-default`,
        short_name: `starter`,
        start_url: `/`,
        background_color: `#663399`,
        theme_color: `#663399`,
        display: `minimal-ui`,
        icon: `src/images/gatsby-icon.png`, // This path is relative to the
                                             root of the site.
      },
    },
  ],
}

```

Now, we will use this styled component in our project. Head over to `layout.js` and update it as follows. We have first removed the import to `layout.css` and added a `GlobalStyle` component. We need to import it from `styled-components`. The updated code is shown in bold in Listing 1-6.

Listing 1-6. `layout.js`

```

import React from "react"
import PropTypes from "prop-types"
import { createGlobalStyle } from 'styled-components'

const Layout = ({ children }) => {
  return (
    <GlobalStyle />
    <main>{children}</main>
  )
}

```



```
const GlobalStyle = createGlobalStyle`  
* {  
  margin: 0;  
  padding: 0;  
  box-sizing: border-box;  
}  
body {  
  font-family: 'Open Sans', sans-serif;  
  color:#262626;  
  background:#fff;  
}  
`
```

```
Layout.propTypes = {  
  children: PropTypes.node.isRequired,  
}
```

```
export default Layout
```

Now, start `gatsby develop`. The site will look like Figure 1-3.

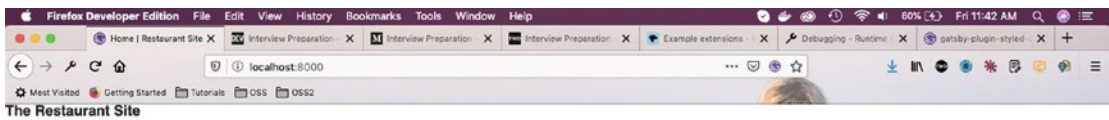


Figure 1-3. localhost

Next, we will install `react-icons` (see <https://www.npmjs.com/package/react-icons>) in the project.

Installing react-icons

To install `react-icons` we must stop `gatsby develop` and then enter the command `npm install --save react-icons`.

Navigate to `index.js` and update the code as shown next, to include an icon for black tie. The updated code is shown in bold in Listing 1-7.

Listing 1-7. index.js

```
import React from "react"
import Layout from "../components/layout"
import { FaBlackTie } from "react-icons/fa"
import SEO from "../components/seo"

const IndexPage = () => (
  <Layout>
    <SEO title="Home" />
    <h3><FaBlackTie />The Restaurant Site</h3>
  </Layout>
)

export default IndexPage
```

It is displayed perfectly in localhost (Figure 1-4).

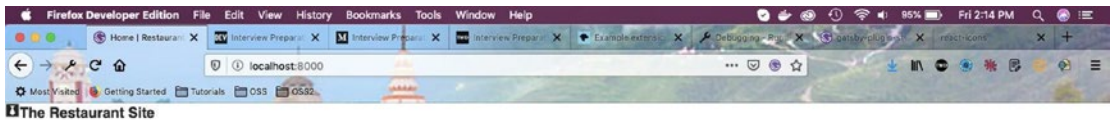


Figure 1-4. localhost

Our setup is complete.

Summary

In this chapter, we learned to create a new Gatsby project, then manage default files, and finally install react-icons into the project.

In the next chapter, we will start creating real parts of the project.

CHAPTER 2

Adding Core Features to the Ecommerce Site

In this chapter we are going to create most of our Gatsby site, which includes the Navbar, the images, banner text, buttons, and the footer section. We are also going to deploy it to Netlify and show a nice Gallery section.

Navbar

We will start by creating the Navbar.

Basic Setup

Create a folder `globals` inside the `components` folder, and then add a `navbar` folder in it. Then create four files inside the `navbar` folder: `Navbar.js`, `NavbarIcons.js`, `NavbarLinks.js`, and `NavbarHeader.js`.

Put the content shown in Listing 2-1 in the `Navbar.js` file.

Listing 2-1. New file `Navbar.js`

```
import React, { Component } from 'react'
import NavbarHeader from './NavbarHeader'
import NavbarLinks from './NavbarLinks'
import NavbarIcons from './NavbarIcons'

class Navbar extends Component {
  render() {
    return (
```

```
        <nav>
          <NavbarHeader />
          <NavbarLinks />
          <NavbarIcons />
        </nav>
      )
    }
  }
}

export default Navbar
```

Next, add the following content in the `NavbarIcons.js` file. We are adding only the basic content now, as in shown in Listing 2-2.

Listing 2-2. New file `NavbarIcons.js`

```
import React, { Component } from 'react'

class NavbarIcons extends Component {
  render() {
    return (
      <div>
        component NavbarIcons
      </div>
    )
  }
}

export default NavbarIcons
```

Next, add the content shown in Listing 2-3 in the `NavbarLinks.js` file.

Listing 2-3. New file `NavbarLinks.js`

```
import React, { Component } from 'react'

class NavbarLinks extends Component {
  render() {
    return (
```

```

    <div>
      component NavbarLinks
    </div>
  )
}
}

export default NavbarLinks

```

Next, put the content shown in Listing 2-4 into the `NavbarHeader.js` file.

Listing 2-4. New file `NavbarHeader.js`

```

import React, { Component } from 'react'

class NavbarHeader extends Component {
  render() {
    return (
      <div>
        component NavbarHeader
      </div>
    )
  }
}

export default NavbarHeader

```

Now, because all the components are created, we will see the Navbar in the layout.js file. The updated code is shown in bold in Listing 2-5.

Listing 2-5. Updating `layout.js` to show Navbar

```

import React from "react"
import PropTypes from "prop-types"
import { createGlobalStyle } from 'styled-components'
import Navbar from "./globals/navbar/Navbar"

const Layout = ({ children }) => {
  return (

```

```

    <GlobalStyle />
    <Navbar />
    <main>{children}</main>
  )
}
...

```

Our Navbar is showing perfectly in localhost (Figure 2-1).



Figure 2-1. localhost

Next, we will add NavWrapper styles for smaller screens in the `Navbar.js` file, by using `styled-components`.

We are also creating a state variable `navbarOpen`, which we are passing as props to the `NavbarLinks` component. We are also creating a function `handleNavbar()`, which has been used to set the state. We are passing it to the `NavbarHeader` component. The updated code is shown in bold in Listing 2-6.

Listing 2-6. Adding styles and state in `Navbar.js`

```

...
import NavbarIcons from './NavbarIcons'
import styled from 'styled-components'

class Navbar extends Component {
  state = {
    navbarOpen: false,
  }

```

```

handleNavbar = () => {
    this.setState({
        navbarOpen: !this.state.navbarOpen
    })
}

render() {
  return (
    <NavWrapper>
    <NavbarHeader handleNavbar={() => this.handleNavbar} />
    <NavbarLinks navbarOpen={this.state.navbarOpen} />
    <NavbarIcons />
    </NavWrapper>
  )
}
}

const NavWrapper = styled.nav`
  @media (min-width: 768px) {
      display: flex;
      align-items: center;
  }
`

export default Navbar

```

I have also added new pictures to the project in the images folder (Figure 2-2). You can take them from the GitHub link at the end of the post.

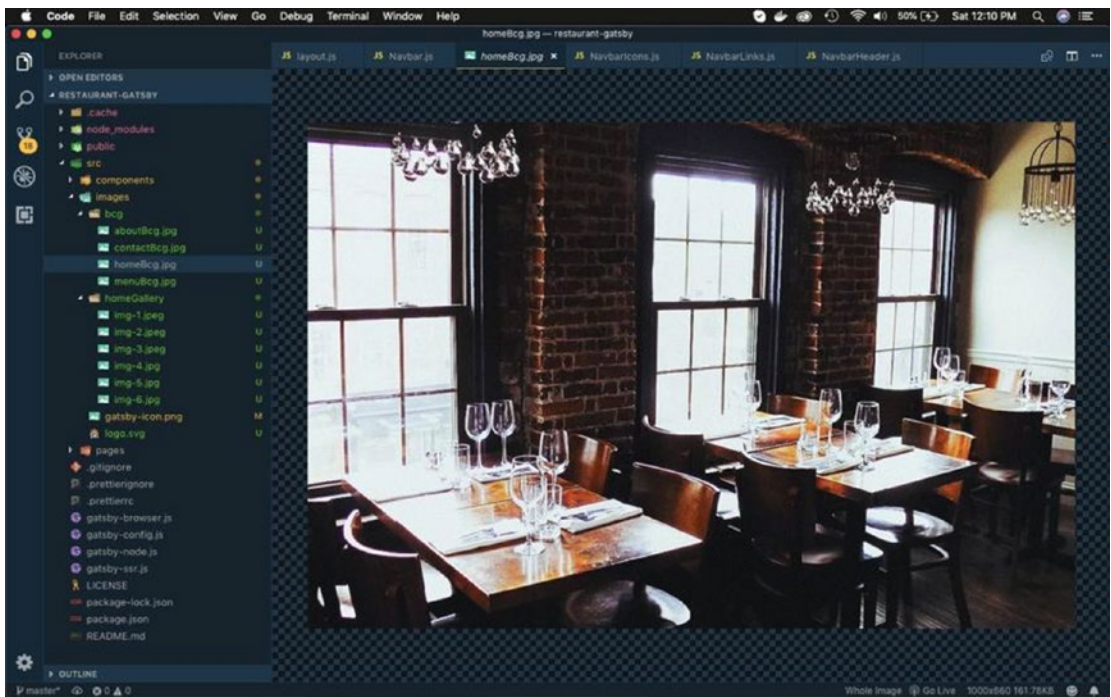


Figure 2-2. Images

NavbarHeader Component

Next, let's complete our NavbarHeader component. Navigate to the NavbarHeader.js file and update the code as shown in Listing 2-7.

Here, we are getting the props handleNavbar and passing it as a callback function when we click on the FaAlignRight icon.

We have also added a styled component HeaderWrapper and inside it given styles for toggle-icon. Notice that we are not displaying the toggle-icon on larger screens.

Listing 2-7. Updating NavbarHeader.js

```
import React from 'react'  
import { Link } from 'gatsby'  
import logo from '../images/logo.svg'  
import { FaAlignRight } from 'react-icons/fa'  
import styled from 'styled-components'
```



```
export default function NavbarHeader({ handleNavbar }) {
  return (
    <HeaderWrapper>
      <Link to="/">
        <img src={logo} alt="company logo" />
      </Link>
      <FaAlignRight className="toggle-icon" onClick={() =>
        {handleNavbar()}} />
    </HeaderWrapper>
  )
}
```

```
const HeaderWrapper = styled.div`
  padding: 0.4rem 1rem;
  display: flex;
  align-items: center;
  justify-content: space-between;
  .toggle-icon {
    font-size: 1.75rem;
    cursor: pointer;
  }
  @media (min-width: 768px) {
    .toggle-icon {
      display: none;
    }
    padding: 0.4rem 1rem;
  }
}
```

Now, our site looks like Figure 2-3 on larger screens.

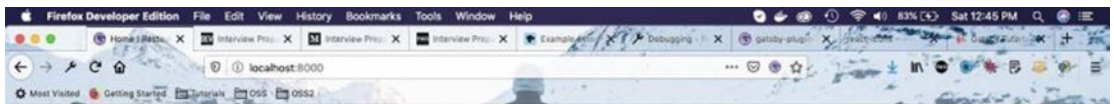


Figure 2-3. Larger screens

Figure 2-4 shows how it will display on smaller screens. Notice that the toggle icon is displayed on this screen.

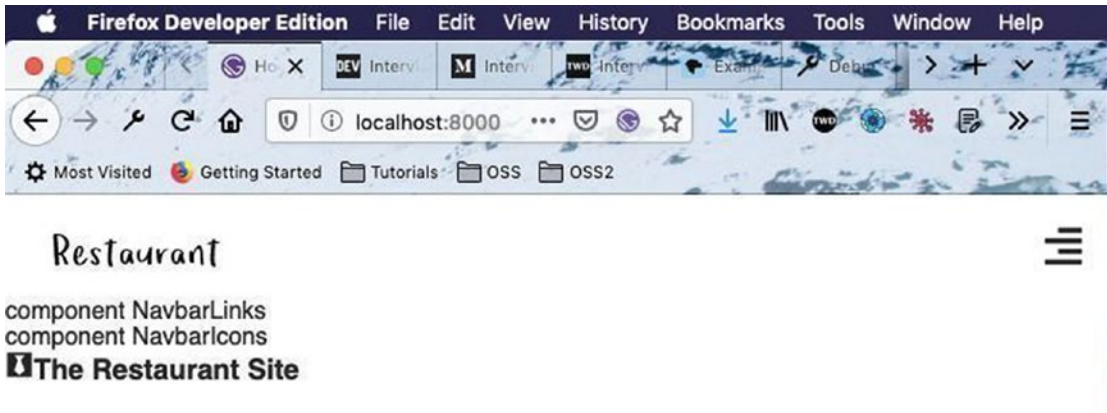


Figure 2-4. *Smaller screens*

We will start creating generic styled components here so that we don't have to create different styles in each project. These styles are also reusable in other parts of the project.

Create a folder named `utils` inside the `src` folder. Create a file named `styles.js` inside the `utils` folder and add the content from Listing 2-8 in the file.

Listing 2-8. New file `styles.js`

```
export const colors = {
  mainWhite: `#fff`,
  mainBlack: `#262626`,
  mainYellow: `#d2aa5c`,
  mainYellow2: `#F2AF29`,
  mainGrey: `#474747`,
}
```

We will soon have other files inside the `utils` folder. Create a root file `index.js` inside it and add the content shown in Listing 2-9 in it.

Listing 2-9. New file `index.js` inside `utils` folder

```
import * as styles from './styles'
export { styles }
```

Let's now use the styles in the `NavbarHeader.js` file. The updated code is shown in bold in Listing 2-10.

Listing 2-10. Using new styles in `NavbarHeader.js`

```
...
import styled from 'styled-components'
import { styles } from '../../utils'

export default function NavbarHeader({ handleNavbar }) {
  return (
    <HeaderWrapper>
      ...
      </HeaderWrapper>
    )
}

const HeaderWrapper = styled.div`
  padding: 0.4rem 1rem;
  display: flex;
  align-items: center;
  justify-content: space-between;
  .toggle-icon {
    font-size: 1.75rem;
    color: ${styles.colors.mainYellow};
    cursor: pointer;
  }
  @media (min-width: 768px) {
    .toggle-icon {
      display: none;
    }
    padding: 0.4rem 1rem;
  }
}
```

Now, the Toggle bar shows the mobile menu in yellow (Figure 2-5).

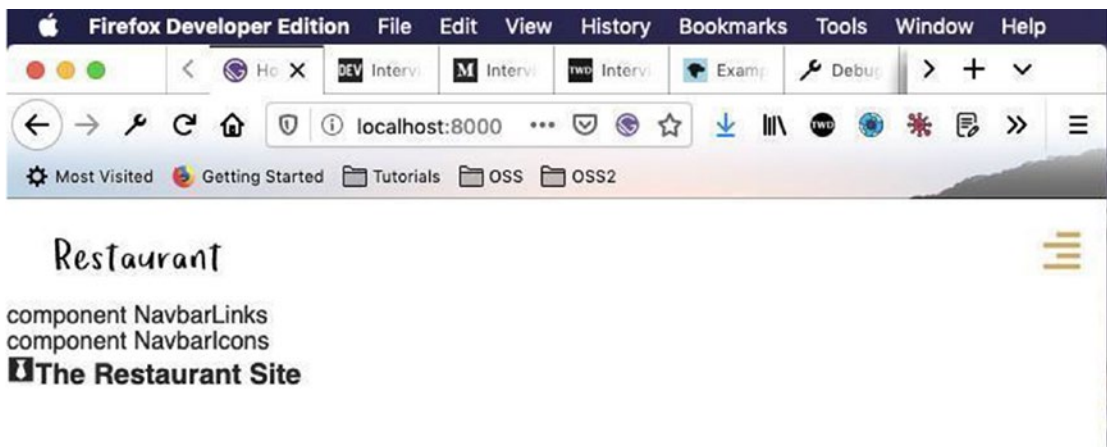


Figure 2-5. Toggle bar

NavBarLinks Component

Next, we will start working on the `NavBarLinks.js` file. We will first add some more imports and a new state variable in the file.

Next, we will render this state variable `links` by using a `map`. We are also using the props `navbarOpen` in the styled component `LinkWrapper`. The code for these is shown in Listing 2-11.

Listing 2-11. Adding state and rendering it in `NavBarLinks.js`

```
import React, { Component } from 'react'
import styled from 'styled-components'
import { Link } from 'gatsby'
import { styles } from '../utils'

class NavBarLinks extends Component {
  state = {
    links: [
      {
        id: 0,
        path: '/',
        name: 'home',
      },
    ],
  }
}
```

```

    {
      id: 1,
      path: '/about/',
      name: 'about',
    },
    {
      id: 2,
      path: '/menu/',
      name: 'menu',
    },
    {
      id: 3,
      path: '/contact/',
      name: 'contact',
    },
  ],
}

render() {
  return (
    <LinkWrapper open={this.props.navbarOpen}>
      {this.state.links.map(item => {
        return (
          <li key={item.id}>
            <Link to={item.path} className="nav-link">
              {item.name}
            </Link>
          </li>
        )
      })}
    </LinkWrapper>
  )
}
}

```

```
const LinkWrapper = styled.ul`  
  `;  
  
export default NavBarLinks
```

It will show our new links on smaller screens (see Figure 2-6), but we need to style them.

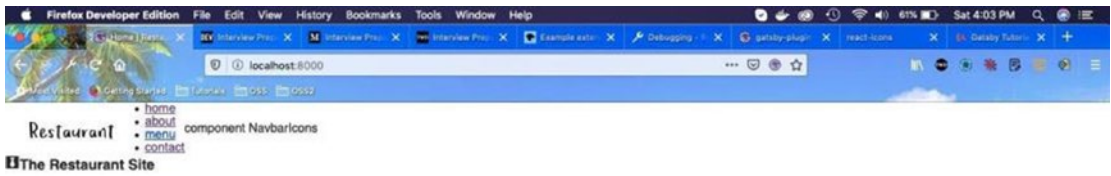


Figure 2-6. Links displayed on a smaller screen

We will add simple styles for the menu including hover style in the styled component LinkWrapper as shown in Listing 2-12.

Listing 2-12. Styles for menu in NavBarLinks.js

```
const LinkWrapper = styled.ul`  
  li {  
    list-style-type: none;  
  }  
  .nav-link {  
    display: block;  
    text-decoration: none;  
    padding: 0.5rem 1rem 0.5rem 1rem;  
    color: ${styles.colors.mainGrey};  
    font-weight: 700;  
    text-transform: capitalize;  
    cursor: pointer;  
    ${styles.transDefault};  
    &:hover {  
      background: ${styles.colors.mainGrey};  
      color: ${styles.colors.mainYellow};  
    }  
  }  
`;
```