Contents

Part 1 – The Server

Part 2 – The Application

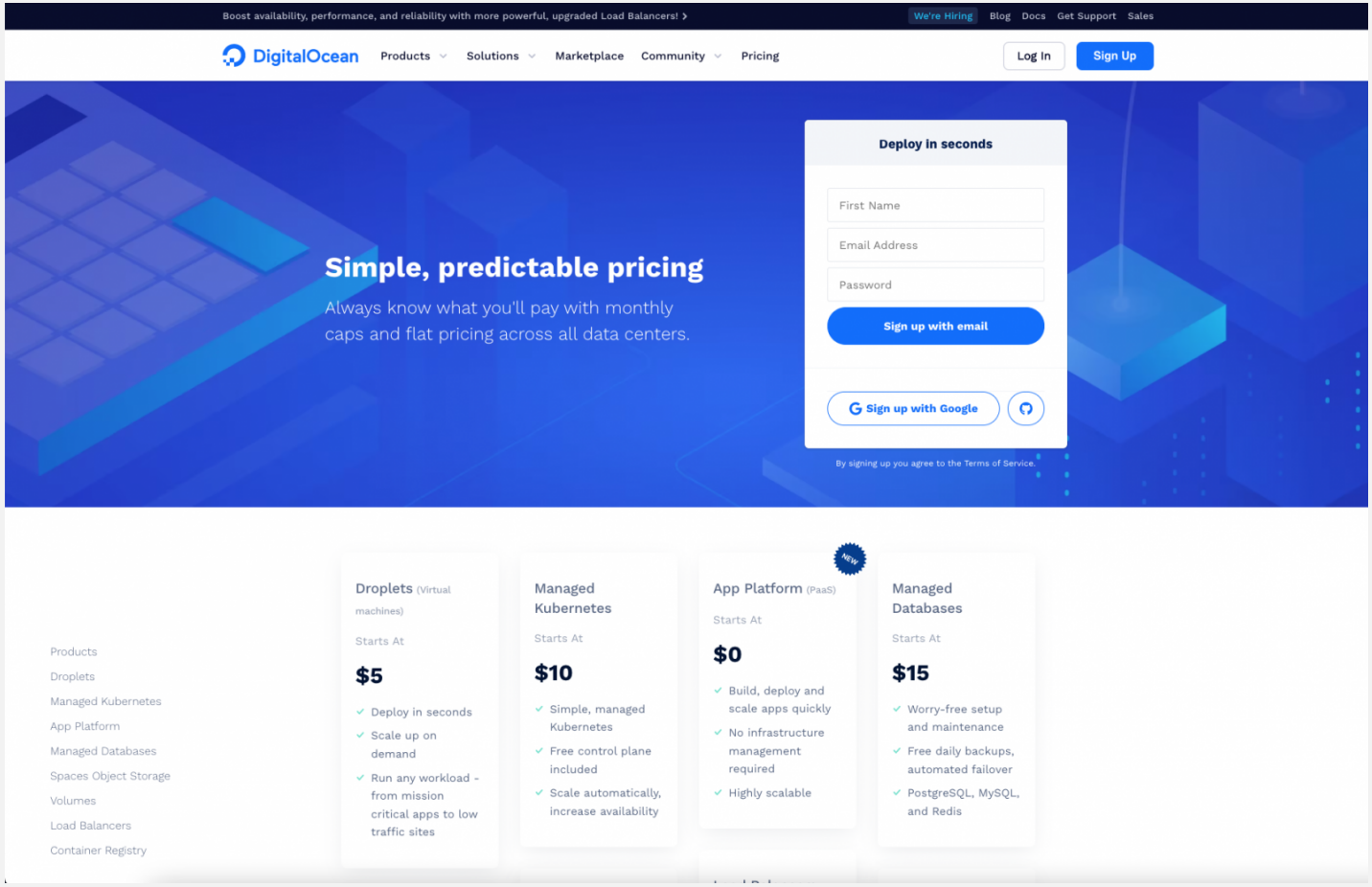
PART 1

The Server

1. Getting a Server

Find a cloud provider that provides a VPS (Virtual Private Server) and signup for an account. A VPS is essentially a virtual machine that is connected to the internet. You can try, for example:

- **https://digitalocean.com** (a VPS is called a droplet here)
- **https://aws.amazon.com/ec2**
- **https://upcloud.com**



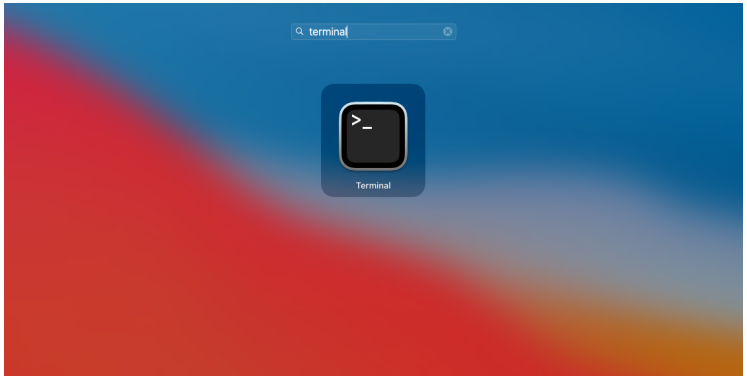
We highly recommed **digitalocean.com** for its ease of use and simple pricing. You can get a server from as low as $5 per month.

After signup you will have the option to choose an operating system. We will use **Ubuntu** (a Linux-based operating system) for our project, so choose the option for **Ubuntu**.

Once your VPS is ready, you will have the information such as server IP and a **root** user password. **Root** user is a special user account that has the highest access rights to your server. That's all that's needed to get started.

2. Connecting to Your Server

To connect to your server, you will need **Terminal** (on Mac) or **Command Prompt** (on Windows). We will use **Terminal** throughout this guide.



Search for 'terminal' on Mac.

Open the Terminal and type SSH command:

```
$ ssh root@<server-ip>
```

Example: ssh root@123.123.123.123

Then enter your password when prompted. You will be connected to your server as a root user and are now ready to manage your server.

SSH stands for **secure shell**. It provides secure connection between your computer and your server.

3. Adding a User

To protect your server, it is good practice to not use the root user on an everyday basis.

1. Now add a user and give it a name, for example: **hobnob**. Specify the password when prompted.

```
$ adduser hobnob
```

The home user directory will be created at: **/home/hobnob**

2. To allow the user to execute commands requiring root privileges, you need to add the user to a group named **sudo**.

```
$ usermod -aG sudo hobnob
```

The **-a** flag appends the user to the group without removing from other group.

3. Then close your terminal and relogin using:

```
$ ssh hobnob@<server-ip>
```

To close your terminal, type **exit** and press Enter/Return.

5. Setting-up a Firewall

With firewall, you can secure your server from unauthorized access by controlling the allowed network traffic to your server. We will use a firewall application named **ufw** (short for uncomplicated firewall)

1. Install ufw.

```
$ sudo apt install ufw
```

2. ufw is inactive by default. To check:

```
$ sudo ufw status
Status: inactive
```

3. View a list of registered applications:

```
$ sudo ufw app list
Available applications:
  OpenSSH
```

You will see that an application named **OpenSSH** is listed. Before activating the firewall, we'll configure the firewall to allow SSH access by allowing **OpenSSH**.

4. Allow SSH access by specifying the application name **OpenSSH**.

```
$ sudo ufw allow OpenSSH
```

5. Now you can enable the firewall:

```
$ sudo ufw enable
```

6. Check the status again:

```
$ sudo ufw status
Status: active

To                         Action      From
--                         ------      ----
OpenSSH                    ALLOW       Anywhere
OpenSSH (v6)               ALLOW       Anywhere (v6)
```

The firewall is now active with rules that allow SSH.

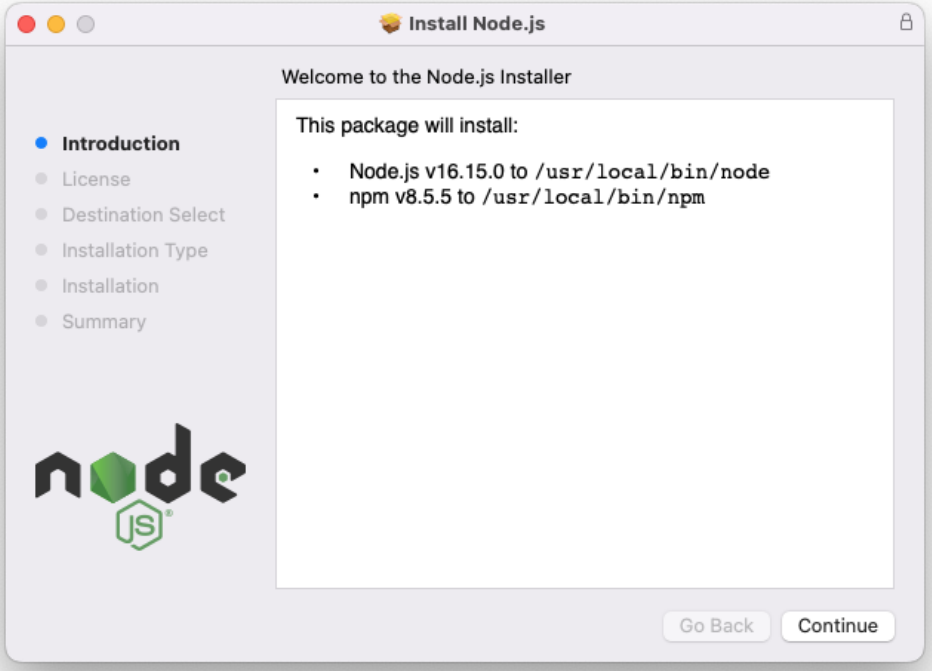
12. Preparing Your Local Computer for Development

In chapter 7, we install and use Node.js to create a website on our server. We also need to have Node.js installed on our local computer for local development.

One of the most common ways to install Node.js is through the official installer. You can download the installer from:

https://nodejs.org/en/download/

Then you can start the installation.



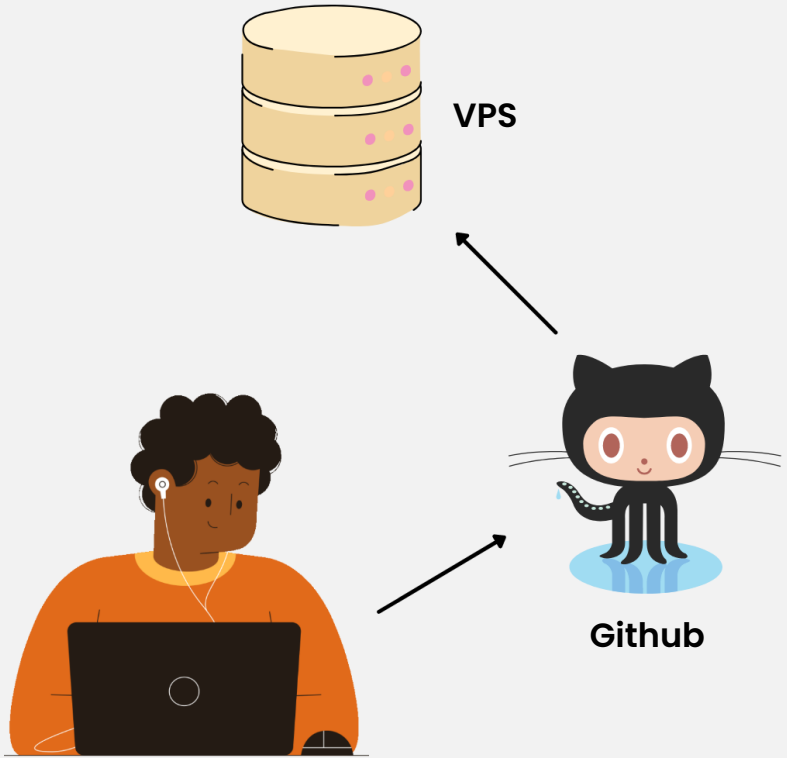
That's all that's needed to start the development. In the next chapter we'll start creating a website on our local computer and then publish the project code on Github.

13. Automating Your Website Publishing

We will setup a project on our local computer and publish the project code on Github. Then we will configure Github to connect to our VPS for automatic deployment and publishing. With this, everytime you push an update from your local computer, your live production website will also get updated.

Steps:

1. Create a New Web Project on Your Local Computer
2. Host Your Web Project on Github
3. Clone Your Github Project to Your VPS
4. Setting Up Automatic Deployment



1. Create a New Web Project on Your Local Computer

1. Open terminal and run this command. We'll create a project named **mywebapp**.

```
$ npx create-next-app mywebapp
```

This will create your website using Next.js framework (https://nextjs.org). Don't worry if you haven't used it before. Just follow the steps to quickly get started and see what it's like.

2. Go to your project folder and run the project:

```
$ cd mywebapp
```

```
$ npm run dev
```

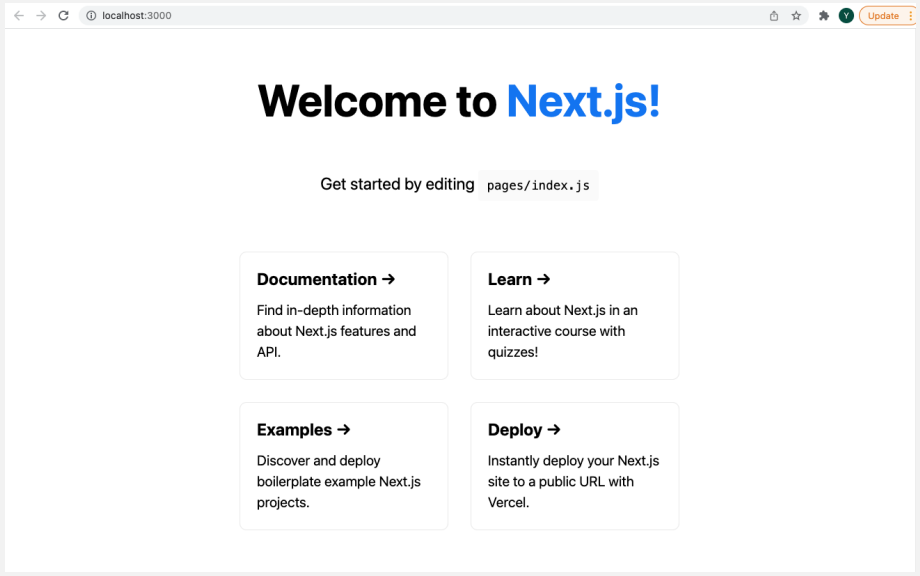
3. Now open your browser and go to:

http://localhost:3000

Congratulations! Your website has been created.

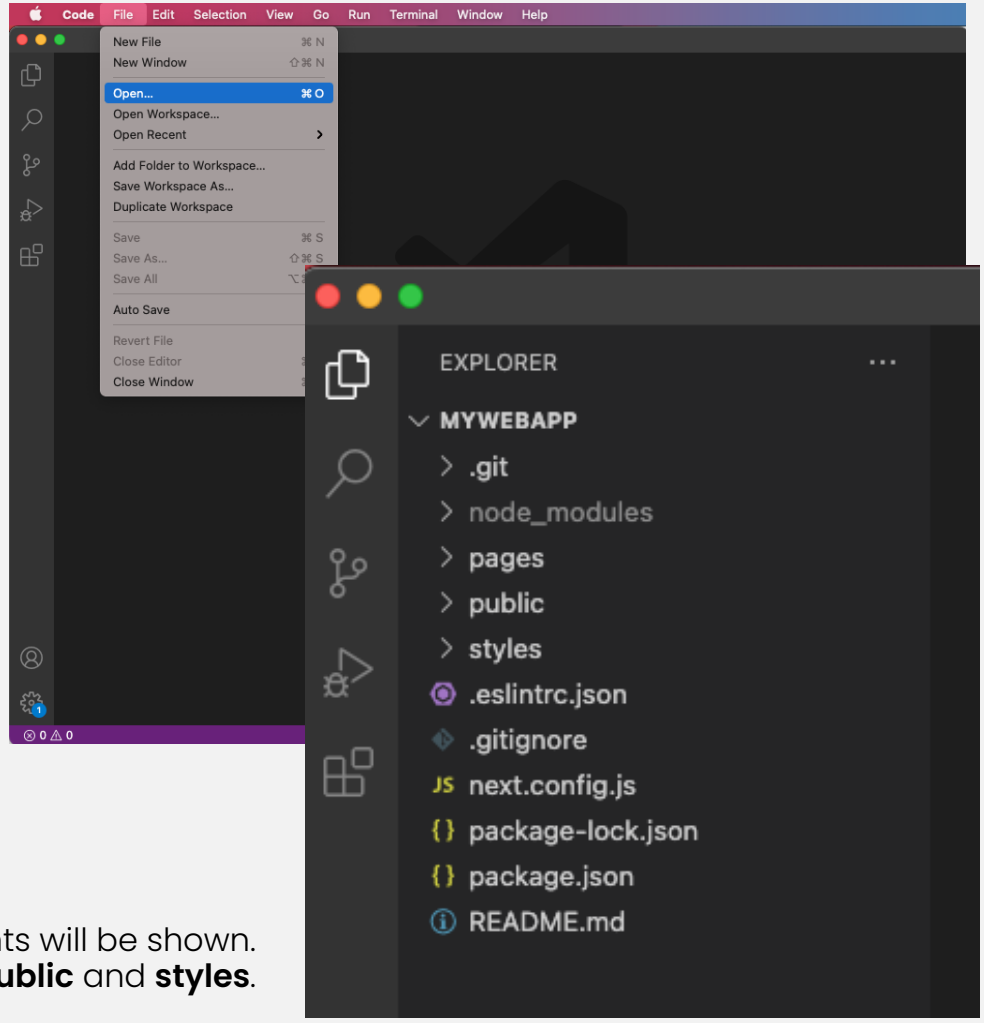
What's so great about Next.js? Next.js is one-of-a-kind framework that allows you to work with both frontend and backend parts of your application. So it is a full-stack framework. It is built on top of the popular React and Node.js, and with its server side rendering feature, your website will be great for SEO. Next.js framework is used by leading companies, including Netflix, Github, Hulu, and AT&T.

This is the default home page of your website:



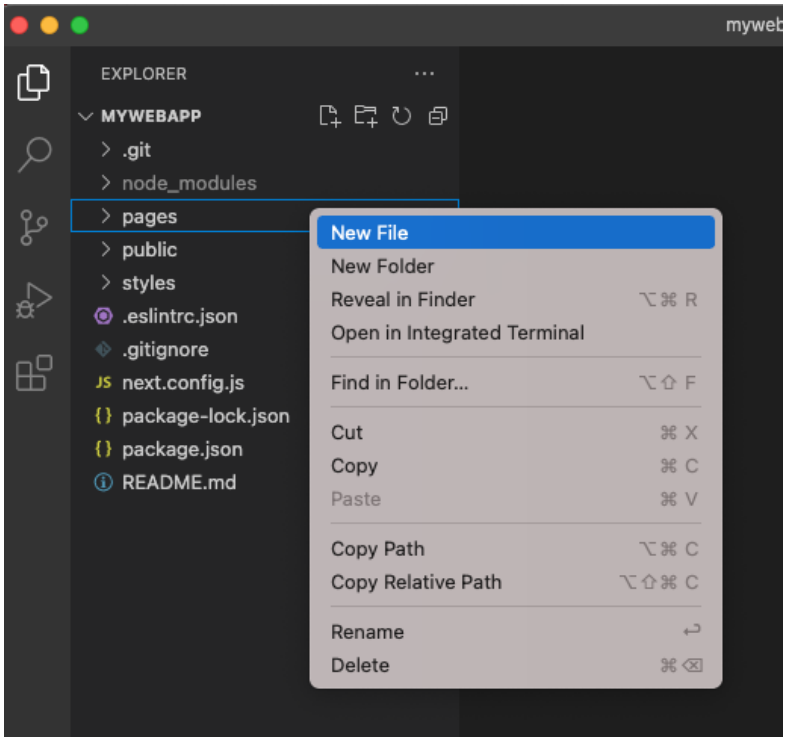
Now, let's try creating a new page. We will use Visual Studio Code as our code editor. You can download Visual Studio Code from https://code.visualstudio.com/download.

4. Open Visual Studio Code. From the **File** menu, select **Open**. Then browse your local computer and select your project folder named **mywebapp**.



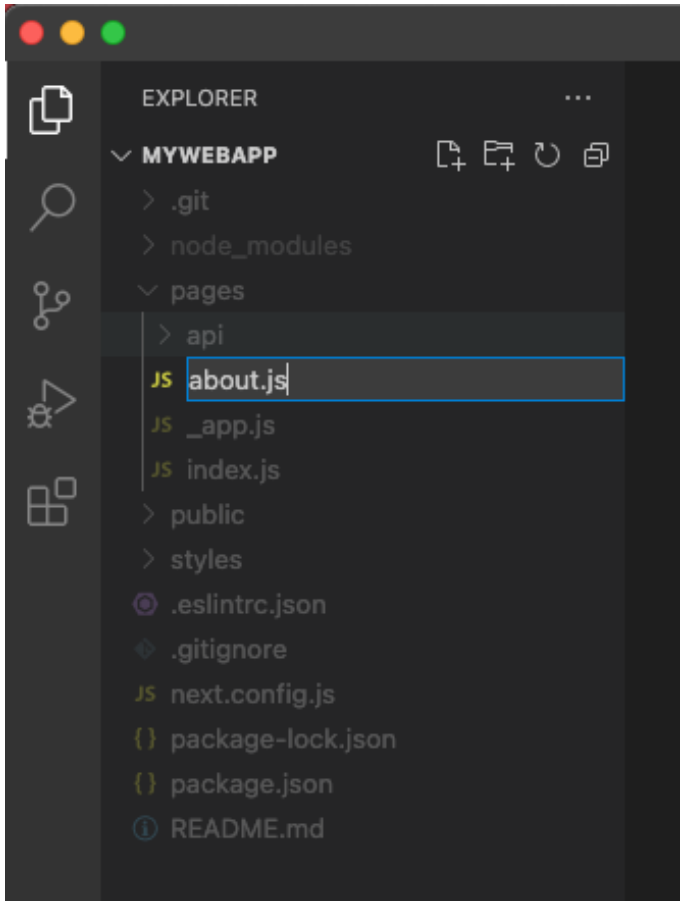
The project contents will be shown.
It starts with 3 folders: **pages**, **public** and **styles**.

5. Create a new file named **about.js** inside the **pages** folder.



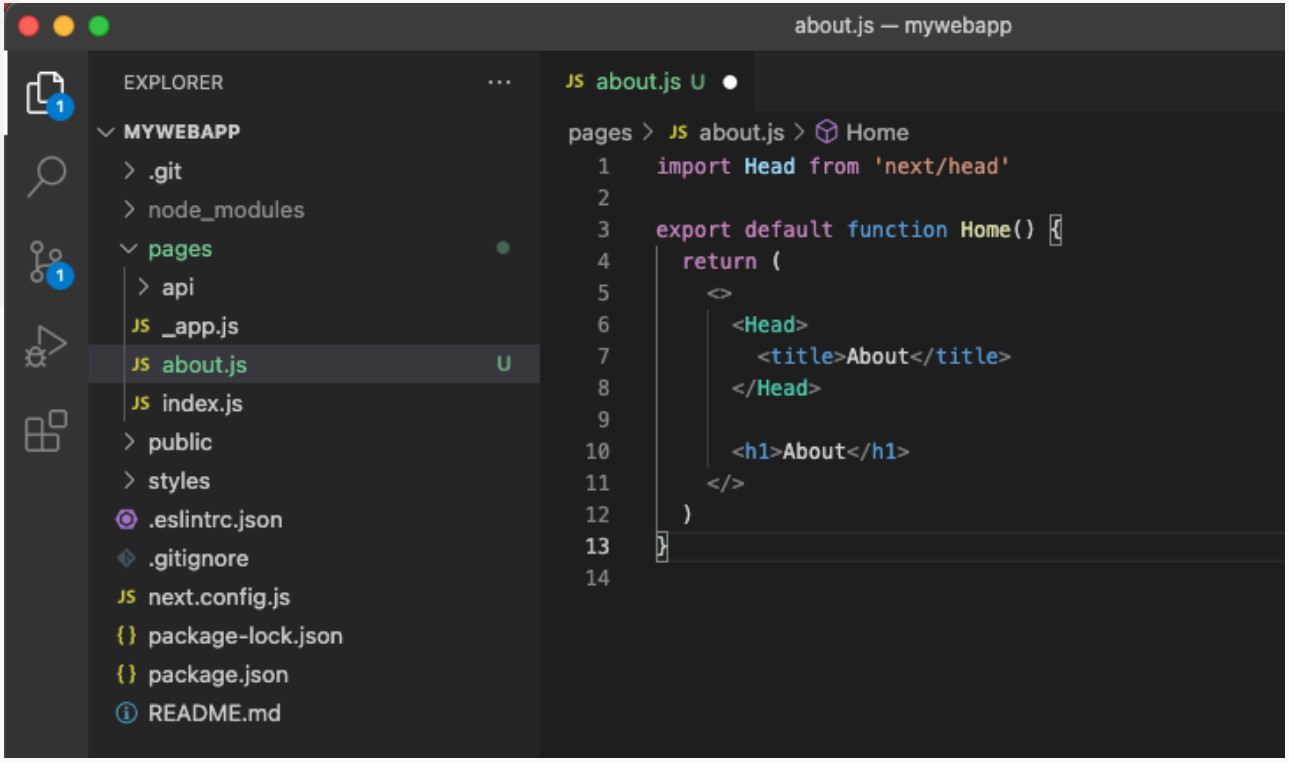
Right click on the **pages** folder and select **New File**.

Enter **about.js** as file name.



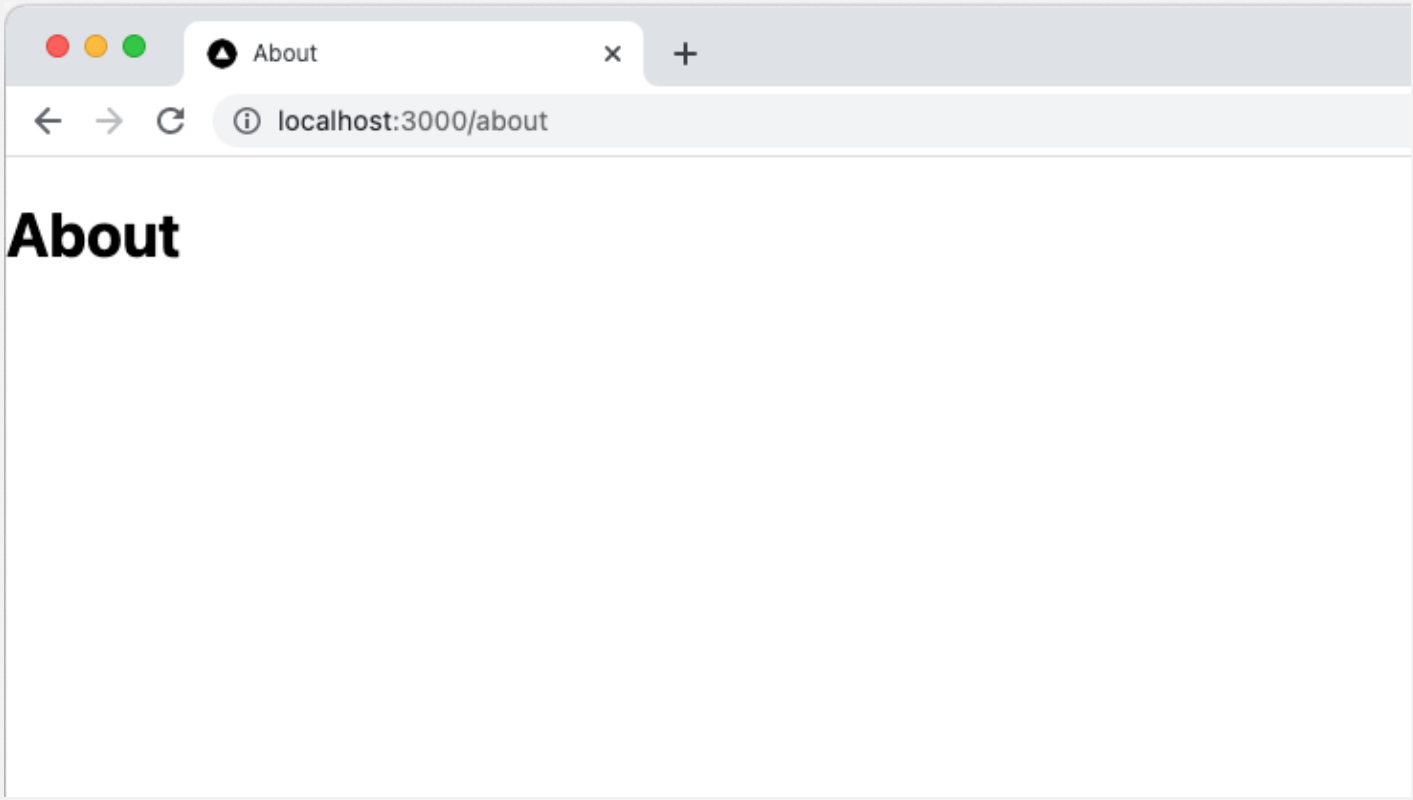
6. Copy the following code and paste it into **about.js**. Then save the file.

```
import Head from 'next/head'

export default function Home() {
  return (
    <>
      <Head>
        <title>About</title>
      </Head>

      <h1>About</h1>
    </>
  )
}
```



Now from your web browser, open:

http://localhost:3000/about

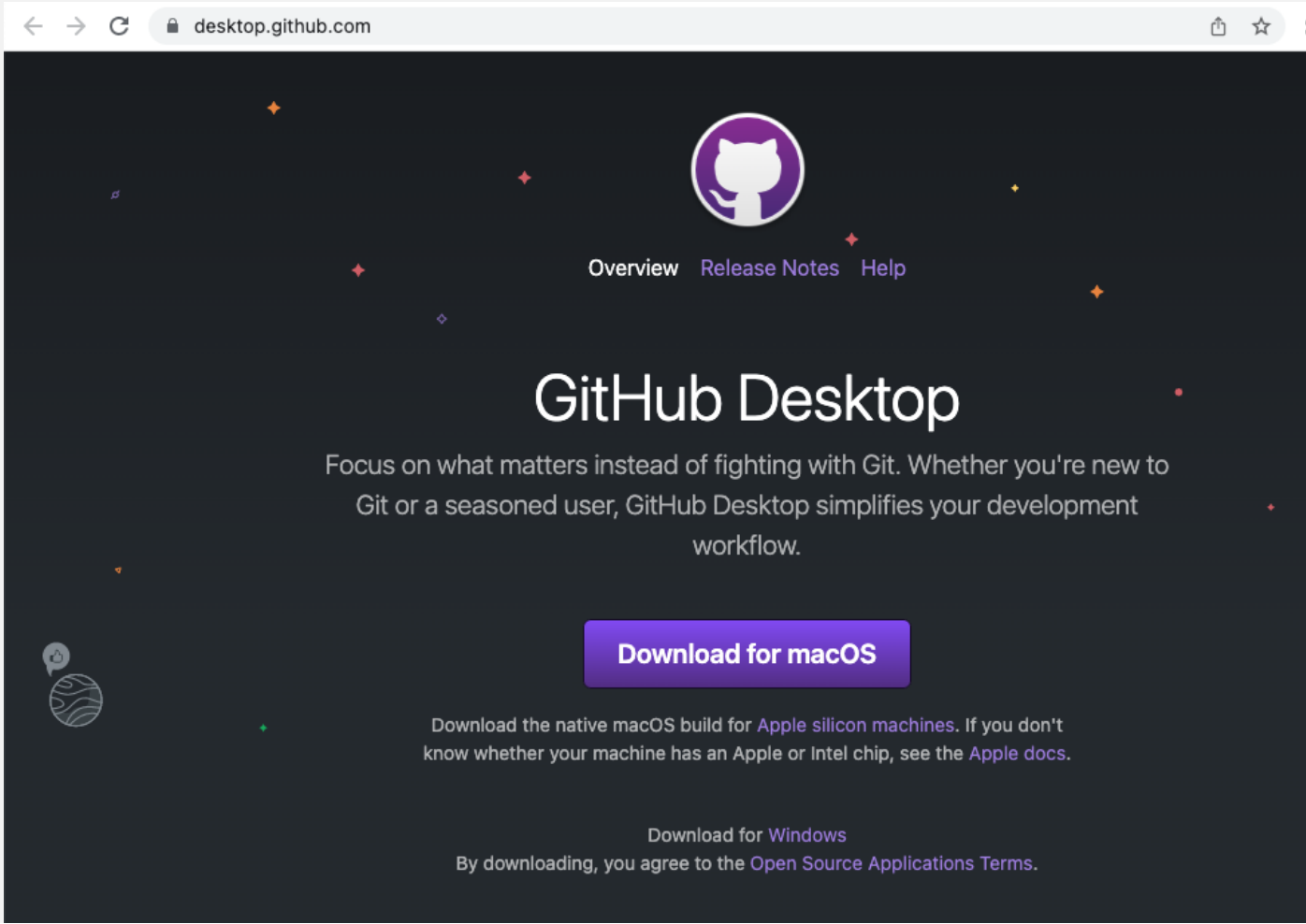


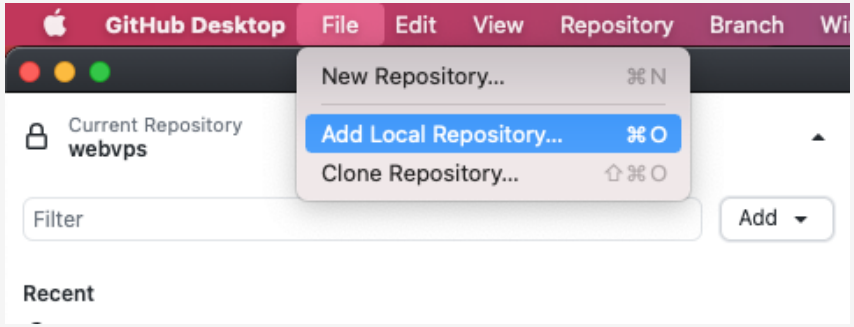
That’s great! Now you have the second page.

2. Host Your Project on Github

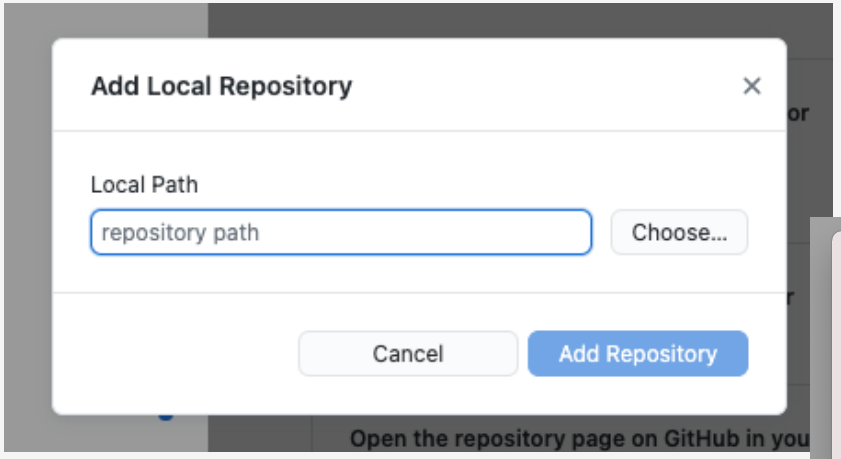
Github (https://github.com) is a code hosting platform. It helps you track code changes and work collaboratively from anywhere. If you don't have a Github account, you can signup for one.

1. To work with Github, we will use Github Desktop. You can download and install Github Desktop from https://desktop.github.com.

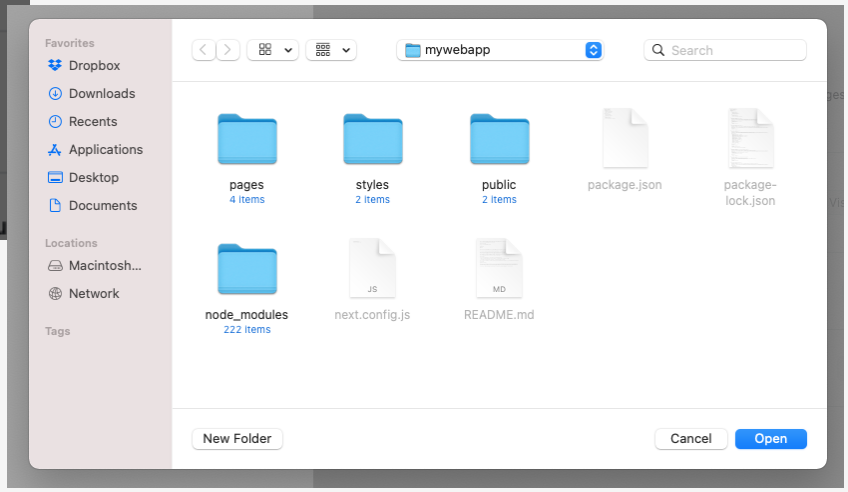


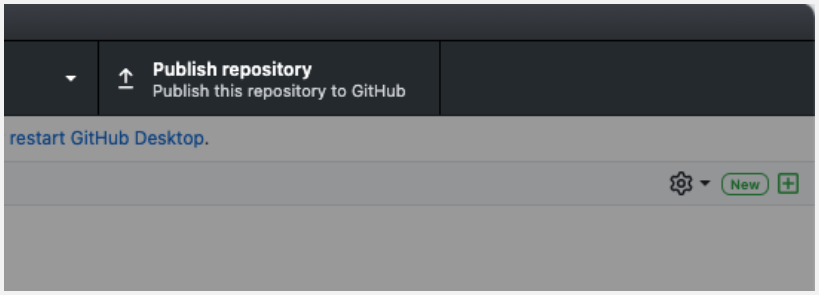


2. Open Github Desktop. From the **File** menu, select **Add Local Repository**.

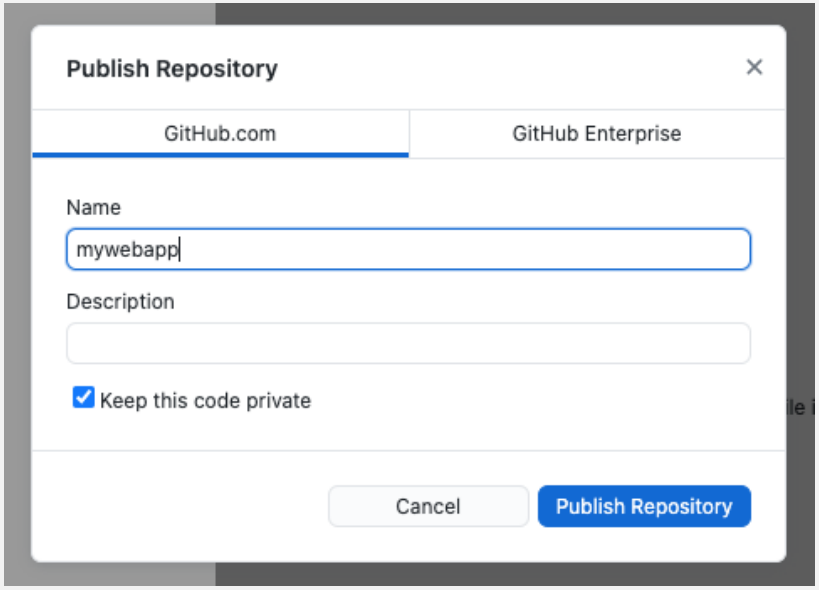


3. Click **Choose**. Browse your local computer to select your project folder **mywebapp** and click **Add Repository**.



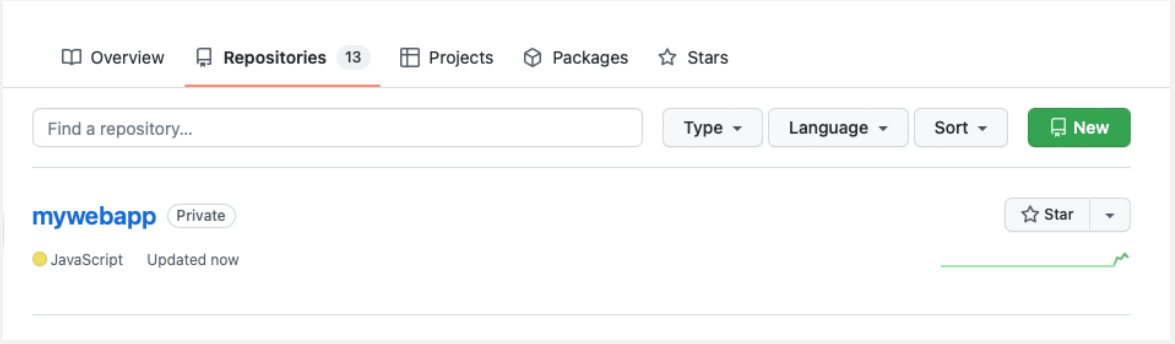


4. Then click **Publish Repository**.



5. A new dialog opens. Here you can enter your project name. Let's make the project private by checking **Keep this code private**. Then click **Publish Repository**.

6. Your project is now hosted on Github.
Open https://github.com and login. Select **Repositories** tab.
You will see your project **mywebapp** is listed.



PART 2

The Application

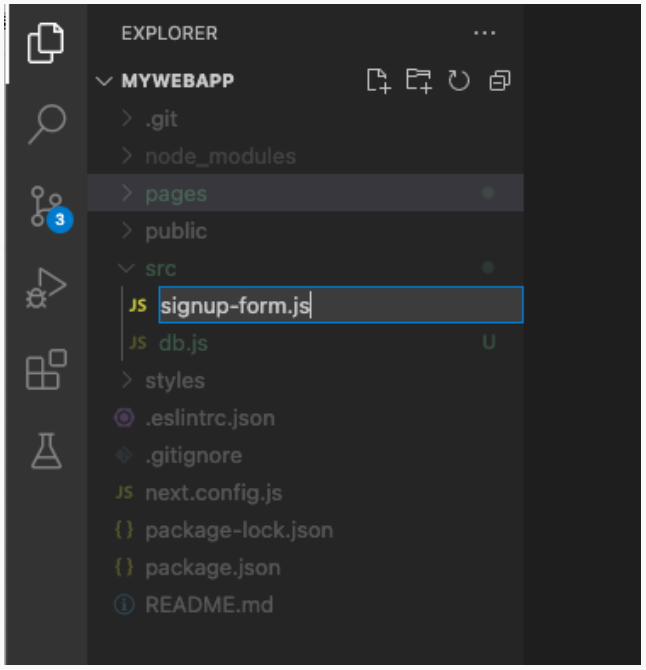
This will be a continuation of our simple Next.js project from Part 1 (chapter 13). In this part, we will enhance our web application to accept user signups and allow users to create their own website.

3. Creating a Component

Remember that how our website is created using Next.js framework? The Next.js framework is built on top of React (client side) and Node.js (server side). Previously, we worked on the server side parts (the database connection, the API route). Now we will work with the client side part, that is the React part.

In this chapter we start with creating a simple **React component**. Don't worry if you haven't used React before. Just follow the steps and you'll see how it works in our application.

1. Create a new js file in the **src** directory. Let's name it **signup-form.js**.



2. Enter the following code.

```
function SignupForm() {
    return <h1>Signup</h1>;
}

export default SignupForm;
```

The keyword **export** in the last line makes the function available to be used in other places in our application

This function will display a title using <h1> element and that's it! You have created a simple component that is available for use in our application.

11. Adding Page Builder

In this chapter, we will allow users to edit their home page.
We will use ContentBox.js, a page builder Javascript library.

3. Adding Page Builder

1. Open terminal, go to your project folder and install **ContentBox.js** library.

```
$ cd mywebapp
```

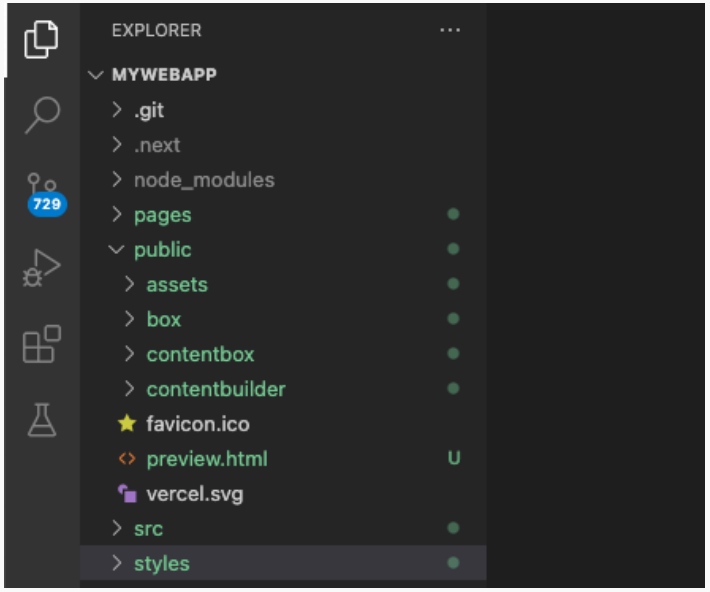
```
$ npm i @innovastudio/contentbox
```

2. Rerun the project.

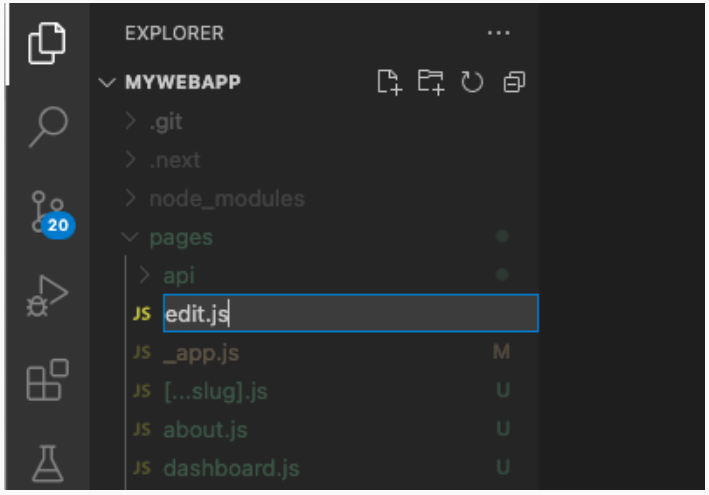
```
$ npm run dev
```

3. From ContentBox.js package copy the following into **public/** folder in your project.

- assets
- box
- contentbox
- contentbuilder
- preview.html



4. Create a new page **pages/edit.js**.



7. To test, open: **http://localhost:3000/edit**.

The ContentBox page builder is now visible. The home page is still empty. You can try adding content by clicking the (+) icon from the left sidebar.

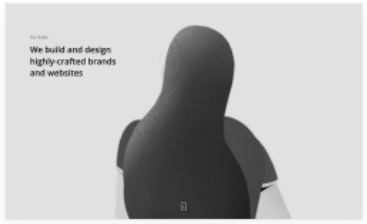
localhost:3000/edit

SIMPLE START:

RANDOM | BASIC | SLIDER | MORE

QUICK START:

HEADER | ARTICLE | MORE



Desgining and building highly-crafted brands and websites.

Nav Bar

One. Two. Three.

LOVE IT

Branding

UI/UX

Front-end

01 02 03 04

Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book.

Introduction

Hi there, I design & build highly-crafted brands & websites

8. Try adding a section from the template. During editing, the page will perform auto saving.