<https://www.gatsbyjs.com/docs/tutorial/getting-started/part-1/#:~:text=To%20create%20a%20new%20Gatsby,web%20browser%20at%20localhost%3A8000%20>.

**Part 1: Create and Deploy Your First Gatsby Site**

**TABLE OF CONTENTS**

* [Introduction](https://www.gatsbyjs.com/docs/tutorial/getting-started/part-1/#introduction)
* [Create a Gatsby site](https://www.gatsbyjs.com/docs/tutorial/getting-started/part-1/#create-a-gatsby-site)
* [Run your site locally](https://www.gatsbyjs.com/docs/tutorial/getting-started/part-1/#run-your-site-locally)
* [Set up a GitHub repo for your site](https://www.gatsbyjs.com/docs/tutorial/getting-started/part-1/#set-up-a-github-repo-for-your-site)
* [Build your site with Gatsby Cloud](https://www.gatsbyjs.com/docs/tutorial/getting-started/part-1/#build-your-site-with-gatsby-cloud)
* [Summary](https://www.gatsbyjs.com/docs/tutorial/getting-started/part-1/#summary)
  + [Key takeaways](https://www.gatsbyjs.com/docs/tutorial/getting-started/part-1/#key-takeaways)
  + [What's coming next?](https://www.gatsbyjs.com/docs/tutorial/getting-started/part-1/#whats-coming-next)

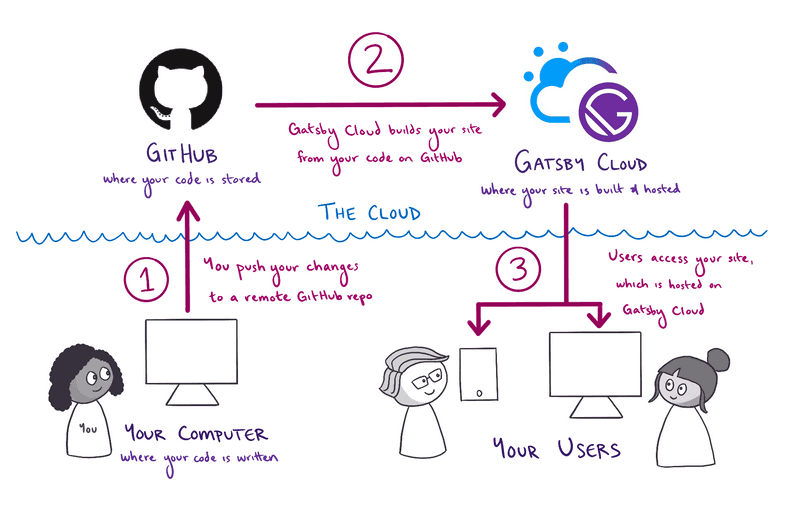
**Introduction**

Now that you’ve set up your computer with all the tools you’ll need, it’s time to get started!

Over the course of this Tutorial, you’ll build and deploy your first Gatsby site: a blog site with support for images and MDX! (If that doesn’t mean anything to you now, that’s okay! It will by the time you reach the end.) Here’s a [finished example](https://gatsbytutorialexample.gatsbyjs.io/) of the site you’ll build. You can also follow along with the [GitHub repository for the finished example](https://github.com/gatsbyjs/tutorial-example).

In this part of the Tutorial, you will go through the process of creating the template for your blog site and deploying it online for everyone to see.

The diagram below shows a high-level view of how all the pieces of this process fit together. (Don’t worry if this doesn’t make sense yet. You’ll learn about each step as you go.)

[](https://www.gatsbyjs.com/static/0fd27b745c1de708f034eaf97c4416e0/d61c2/deployment-workflow.png)

*Expand for detailed description*

**Prefer a video?**

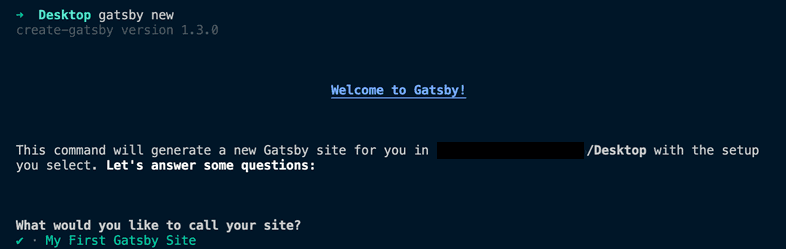
If you’d rather follow along with a video, here’s a recording of a livestream that covers all the material for Part 1.

**Note**: Parts of this recording may be slightly outdated, but the concepts are generally applicable. For the most up-to-date information, follow along with the written tutorial.

Don’t want to miss any future livestreams? Follow our [Gatsby Twitch channel](https://www.twitch.tv/gatsbyjs).

**Create a Gatsby site**

To create your first Gatsby site, you’re going to use a command from the Gatsby command line interface (CLI): gatsby new. This command brings up an interactive prompt that asks you questions about the site you want to build. After you enter all the information, the CLI uses your answers to automatically generate your new Gatsby site.

[](https://www.gatsbyjs.com/static/eab322d4f0a5a12bdc749ef0992c4e7c/e92cd/gatsby-new-cli.png)

**Note:** For this Tutorial, your Gatsby CLI should be v4.8 or newer. To check what version you have installed, run the following command:

Copycopy code to clipboard

gatsby --version

Need to update? Run the command below to get the latest version of the Gatsby CLI:

Copycopy code to clipboard

npm install -g gatsby-cli

Let’s take a closer look at the process:

1. Open the command line, and use the cd command to change directories into the folder where you want to create your new Gatsby site. For example, if you wanted to create your new site on your desktop, you might type:

Copycopy code to clipboard

cd Desktop

1. Run the following command from the command line. This will start up the interactive prompt to help you create a new Gatsby site.

Desktop

CopyDesktop: copy code to clipboard

gatsby new

**Having trouble with**gatsby new**?** If you had trouble globally installing gatsby-cli in Part 0, you can also create a new site by running npm init gatsby from the command line instead of gatsby new.

1. When the prompt asks, **“What would you like to call your site?”** enter a name for your site.

Copycopy code to clipboard

What would you like to call your site?

✔ · My First Gatsby Site

1. When the prompt asks, **“What would you like to name the folder where your site will be created?”** use the default folder name, which will be based on the site name you chose.

Copycopy code to clipboard

What would you like to name the folder where your site will be created?

✔ Desktop/ my-first-gatsby-site

1. When the prompt asks, **“Will you be using JavaScript or TypeScript?”** choose **JavaScript**.

Copycopy code to clipboard

Will you be using JavaScript or TypeScript?

❯ JavaScript

TypeScript

This tutorial doesn’t require any prior TypeScript knowledge as it uses JavaScript. If you’re familiar with TypeScript you can read the [Gatsby and TypeScript guide](https://www.gatsbyjs.com/docs/how-to/custom-configuration/typescript/) to learn about typings, files, and conventions. If you want to use TypeScript we recommend going through the tutorial first and then only afterwards convert the project to TypeScript.

1. When the prompt asks, **“Will you be using a CMS?”** select **“No (or I’ll add it later)”**.

Copycopy code to clipboard

✔ Will you be using a CMS?

· No (or I'll add it later)

In the future, you can use these options to tell gatsby new what features you want to add to your site, and gatsby new will automatically configure them for you. It’s a much quicker way to set up new projects.

But in this first site, you’ll set things up manually to learn about how Gatsby’s pieces fit together.

1. When the prompt asks, **“Would you like to install a styling system?”** select **“No (or I’ll add it later)”**. (You’ll add styles manually later.)

Copycopy code to clipboard

✔ Would you like to install a styling system?

· No (or I'll add it later)

1. When the prompt asks, **“Would you like to install additional features with other plugins?”** use the arrow and Enter keys to select **“Done”**.

Copycopy code to clipboard

✔ Would you like to install additional features with other plugins?

· Done

1. The prompt will show you a summary of what gatsby new will do. It should look something like the output below.

Copycopy code to clipboard

Thanks! Here's what we'll now do:

🛠 Create a new Gatsby site in the folder my-first-gatsby-site

? Shall we do this? (Y/n) › Yes

1. When the prompt asks, **“Shall we do this?”** enter **“Y”**. The gatsby new command will start building your site. Your internet download speed will affect how long this command takes to run. After it finishes, you should see a message like this:

Copycopy code to clipboard

🎉 Your new Gatsby site My First Gatsby Site has been successfully

created at ~/Desktop/my-first-gatsby-site.

Start by going to the directory with

cd my-first-gatsby-site

Start the local development server with

npm run develop

See all commands at

https://www.gatsbyjs.com/docs/gatsby-cli/

Congratulations, you’re now the owner of a brand-new Gatsby site!

**Run your site locally**

So far, you’ve generated the code for your site, but what does it actually look like in a web browser like Firefox or Google Chrome? To find out, you’ll first need to start up your site’s local development server.

The **development server** is a useful tool for when you’re working on your site locally (from your own computer). When your development server is running, you can use a web browser to interact with your local copy of your site. That way, you can test out changes to your code, to make sure they work before you actually publish a new version of your site to the internet.

To start up your development server, do the following:

1. In the command line, change into the directory you created for your site:

~/Desktop

Copy~/Desktop: copy code to clipboard

cd my-first-gatsby-site

**Tip:** Whenever you want to run any commands for your site, you need to be in the context for that site. That is, your command line needs to be pointed at the directory where your site’s code lives.

1. From your site directory, start the development server by running the following command:

~/Desktop/my-first-gatsby-site

Copy~/Desktop/my-first-gatsby-site: copy code to clipboard

gatsby develop

If you weren’t able to install the Gatsby command line interface globally, you can start your development server using the following command instead:

Copycopy code to clipboard

npm run develop

1. After a few moments, the command line should output a message like the following, telling you your development server is ready to go!

Copycopy code to clipboard

You can now view my-first-gatsby-site in the browser.

⠀

http://localhost:8000/

⠀

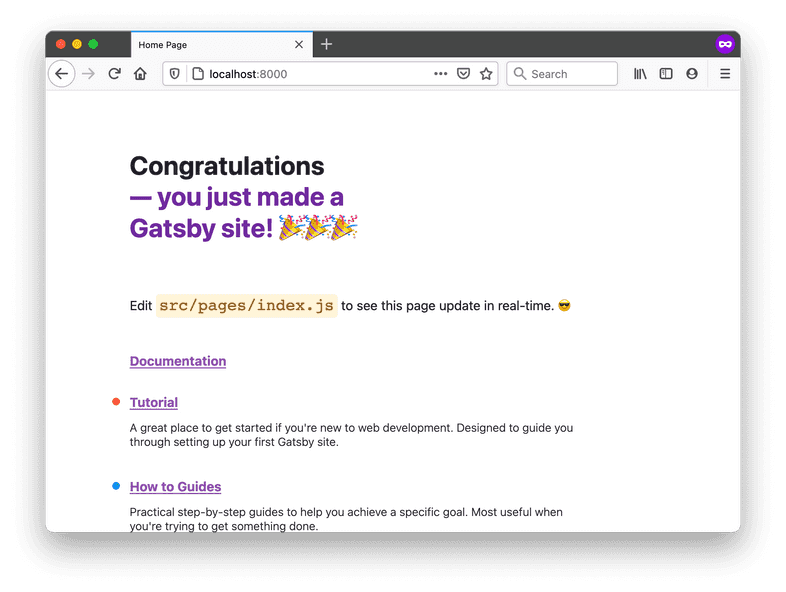
View GraphiQL, an in-browser IDE, to explore your site's data and

schema

⠀

http://localhost:8000/\_\_\_graphql

1. Open your favorite web browser and navigate to http://localhost:8000.

[](https://www.gatsbyjs.com/static/b79cb66545b144295a8c6a5efeaafb20/94cea/localhost-new-site.png)

And there it is: your very first Gatsby site! 🎉

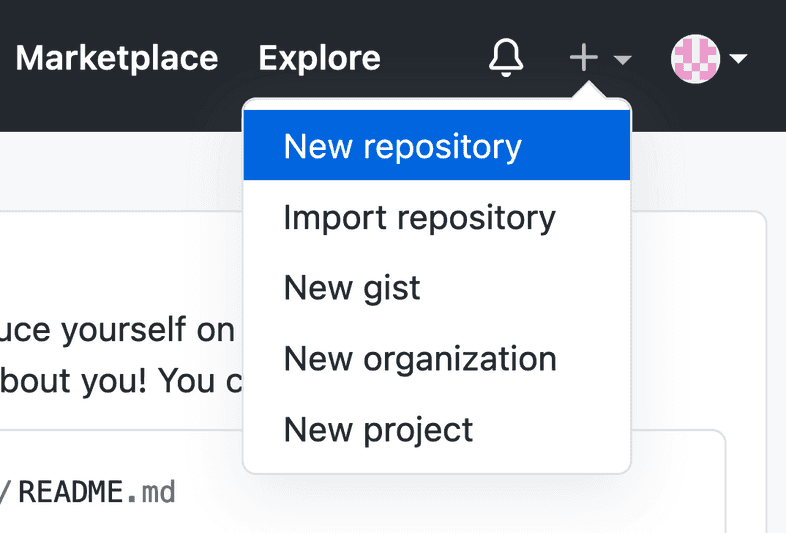
You’ll be able to visit the site locally at http://localhost:8000/ for as long as your development server is running. (That’s the process you started by running the gatsby develop command.) To stop running that process (or to “stop running the development server”), go back to your terminal window, hold down the “control” key, and then hit “c” (ctrl-c). To start it again, run gatsby develop again!

**Note:** If you are using VM setup like vagrant and/or would like to listen on your local IP address, run gatsby develop --host=0.0.0.0. Now, the development server listens on both http://localhost and your local IP.

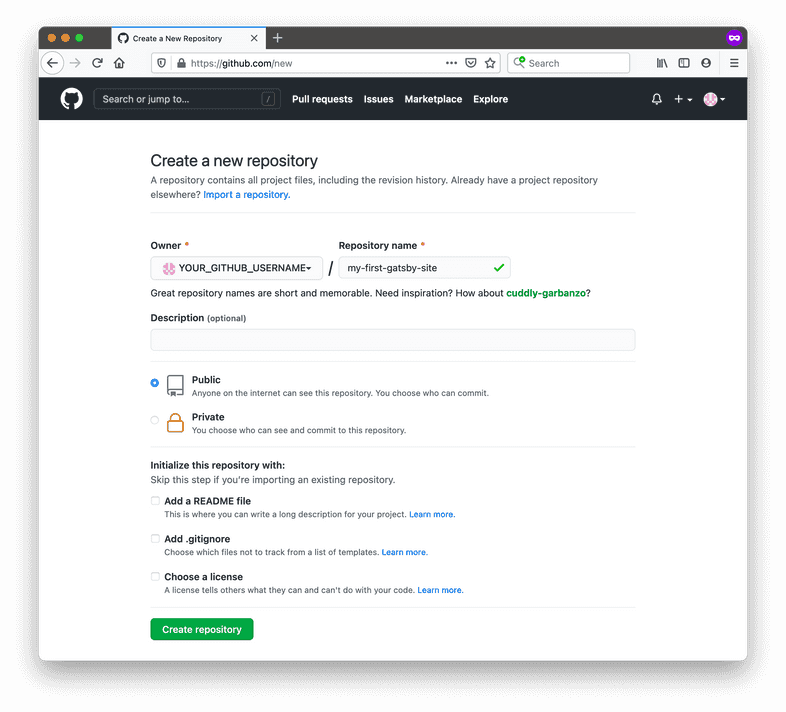
**Set up a GitHub repo for your site**

GitHub is a website that many developers use to back up and share their code online. By uploading your code to GitHub, you’ll be able to work on the same codebase from multiple computers. You’ll also be able to use Gatsby Cloud to build and host your site.

1. Each codebase on GitHub is stored in its own **repository** (also called a “repo”, for short). To create a new repository for your blog, click the plus icon in the top-right corner of the navigation bar. Select “New repository”.

[](https://www.gatsbyjs.com/static/bf74830c88d3f8b0287b58cf397be992/18539/new-repo-button.png)

1. When filling out the form for your new repo, you can make it public or private. (This only affects the visibility of your code on GitHub. Your site will still be visible to everyone once you deploy it with Gatsby Cloud.) Leave the initialization option checkboxes unchecked.

[](https://www.gatsbyjs.com/static/94ec685d2adefdf4d2aac5b3364acba9/3d68f/new-repo-options.png)

1. To push your existing code from your computer to your new GitHub repository, enter the commands below in the command line. Be sure to swap out YOUR\_GITHUB\_USERNAME for your actual username and YOUR\_GITHUB\_REPO\_NAME with the name you gave your GitHub repo (like my-first-gatsby-site).

Copycopy code to clipboard

git remote add origin https://github.com/YOUR\_GITHUB\_USERNAME/YOUR\_GITHUB\_REPO\_NAME.git

git branch -M main

git push -u origin main

**Using GitHub for the first time?**

If you get an error about permissions when you try to push your code to GitHub for the first time, you might need to set up a **personal access token** for your GitHub account. This lets GitHub know that your computer has permission to push code changes to your remote repos.

For instructions on how to set up a personal access token, follow GitHub’s guide: [Creating a personal access token](https://docs.github.com/en/authentication/keeping-your-account-and-data-secure/creating-a-personal-access-token). Your personal access token will need the **repo** scope to be able to push changes to your repository.

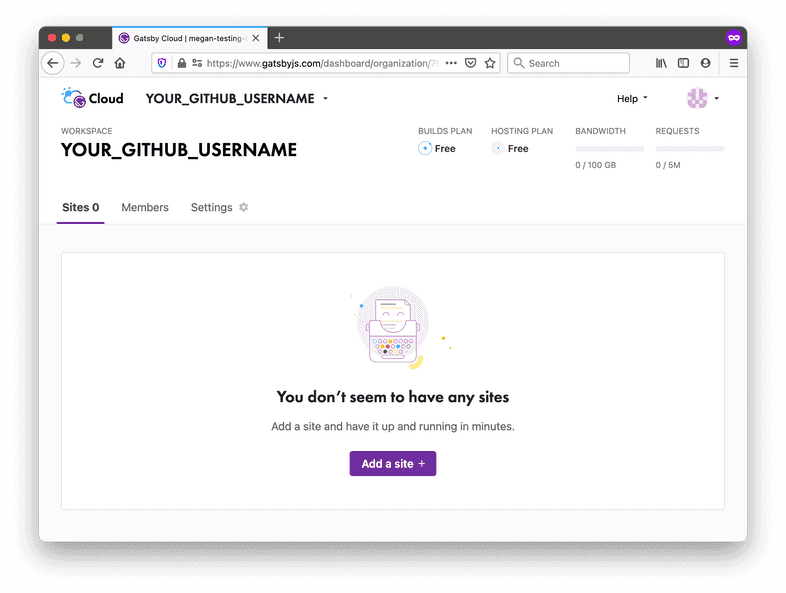
Now you have a copy of your code saved on GitHub’s servers. In the next step, you’ll connect your Gatsby Cloud account to the GitHub repo you just created.

**Build your site with Gatsby Cloud**

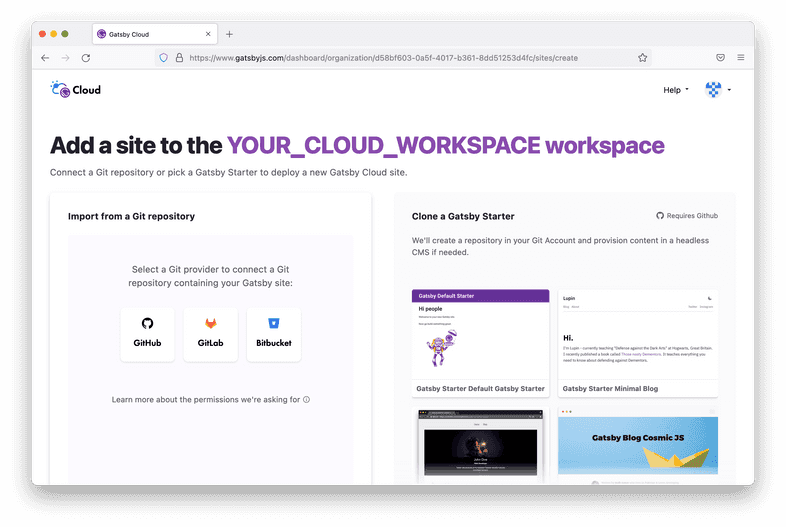
Gatsby Cloud is an infrastructure platform that is specifically optimized for building, deploying, and hosting Gatsby sites. Once you connect your Gatsby Cloud account to your GitHub repository, Gatsby Cloud will build your site and make it available for others on the internet to see.

To connect your code on GitHub to your Gatsby Cloud account, do the following:

1. Go to your [Gatsby Cloud Dashboard](https://www.gatsbyjs.com/dashboard/). Click on the **“Add a site”** button.

[](https://www.gatsbyjs.com/static/9c130998b561f1770834309715c99d5b/2b36a/01-create-a-site-button.png)

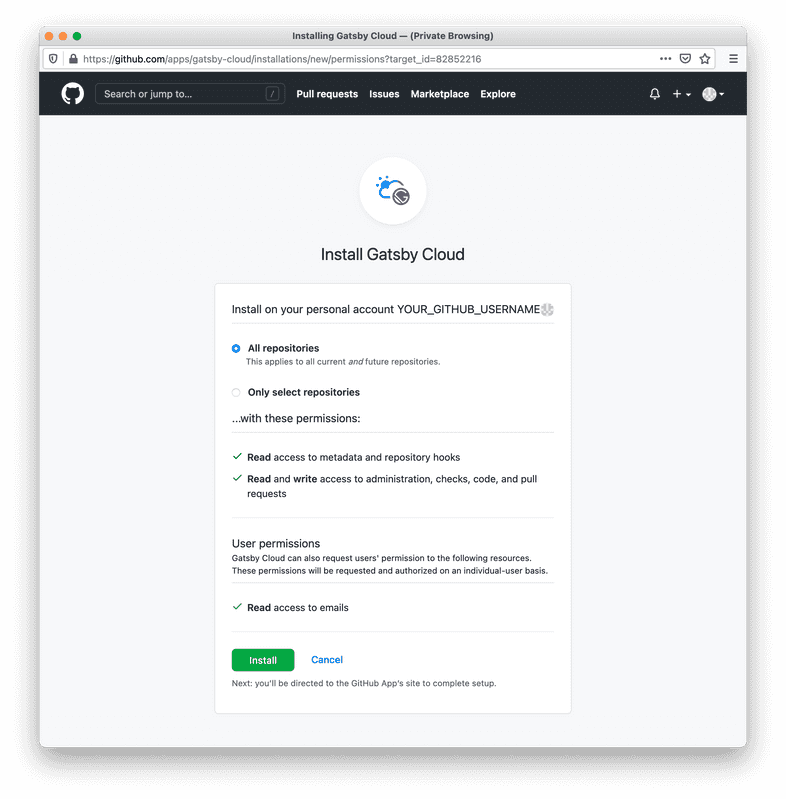
1. The next few steps will help you add your site to Gatsby Cloud. First, in the **“Import from a Git repository”** card click the **“GitHub”** icon to select GitHub as your Git provider.

[](https://www.gatsbyjs.com/static/ff5dd96106160a1c11eaa25af6becdda/0f688/02-import-a-git-repo.png)

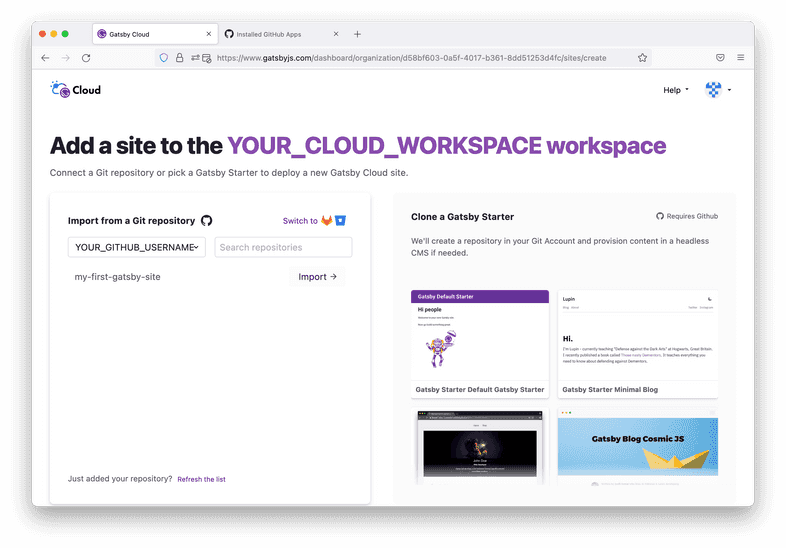
1. If this is your first time connecting GitHub to Gatsby Cloud, you’ll need to give Gatsby Cloud permission to access your GitHub account.

**Note:** If you are part of more than one GitHub organization, you will need to first select the organization with which the repository resides at this step before selecting the repository itself.

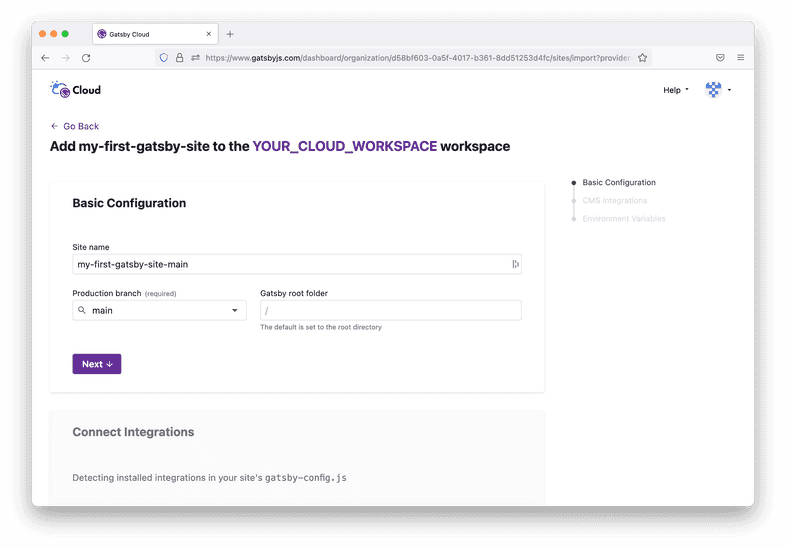
1. A new browser window should open, where GitHub will ask you whether you want to give Gatsby Cloud permission to your GitHub repositories. You can choose whether to give Gatsby Cloud access to all of your GitHub repositories or to only the repository you created (my-first-gatsby-site). Then click **“Install”**.

[](https://www.gatsbyjs.com/static/4fd11cb2e4af910ca099f70d12aa8421/0f96c/03-github-gatsby-cloud-permissions.png)

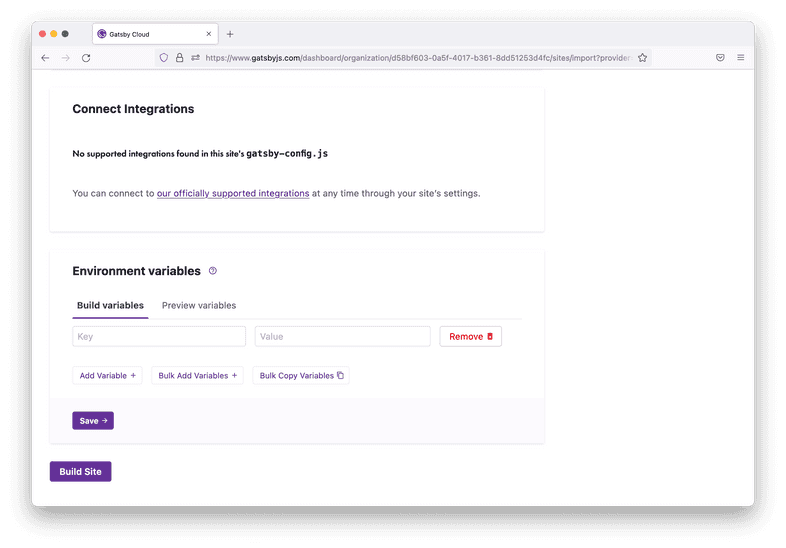
1. Now, when you go back to the Gatsby Cloud window, the repository list should contain your GitHub repository. Select it by clicking on **“Import”**.

[](https://www.gatsbyjs.com/static/5fb2c6c66c2d25426b180ee40917fd83/65781/04-select-repository.png)

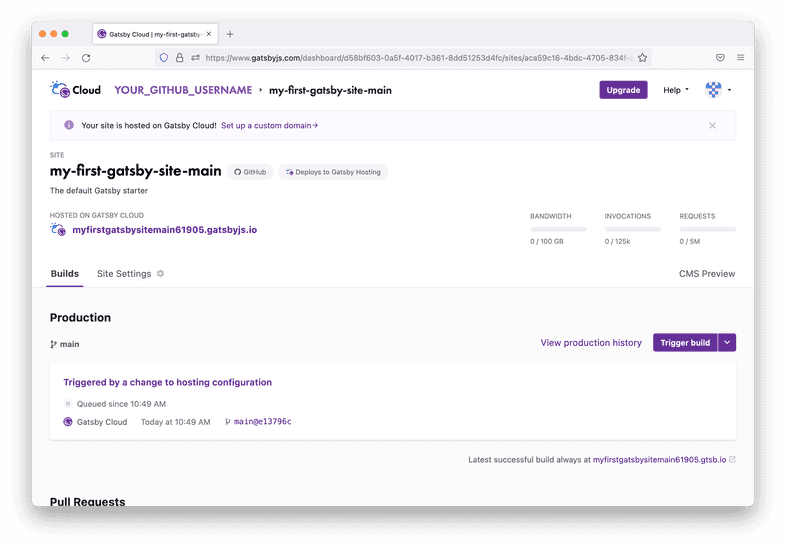
1. Once you select your repo, you’ll be navigated to the configuration step which presents you with a few more inputs. These let you tell Gatsby Cloud where to look in your GitHub repo for your Gatsby site. You can also change what Gatsby Cloud will name your site. **Leave the default settings** and click the **“Next”** button.

[](https://www.gatsbyjs.com/static/61bb418dbf509217b076a19507374eef/65781/05-add-site-details.png)

1. Gatsby Cloud will ask you if you want to add any integrations to your site. For future projects, this might be useful if you want to use a CMS. Gatsby Cloud will also ask if you want to add any environment variables. Again, this may be useful for future projects, but for now, scroll past and click the **“Build Site”** button.

[](https://www.gatsbyjs.com/static/c36f2eede71bb383cf02e73a7a8cf320/65781/06-integrations-and-environment-variables.png)

1. Now that your site has been created, you’ll be taken to a site dashboard where you can see the status of your builds. Gatsby Cloud should start building your site automatically. You’ll see a link to your new site, which is automatically hosted on Gatsby Cloud. You can share this link with anyone, and they’ll be able to see your site online!

[](https://www.gatsbyjs.com/static/d82ecf06f74d4195697a9a4c9253049d/65781/07-site-page.png)

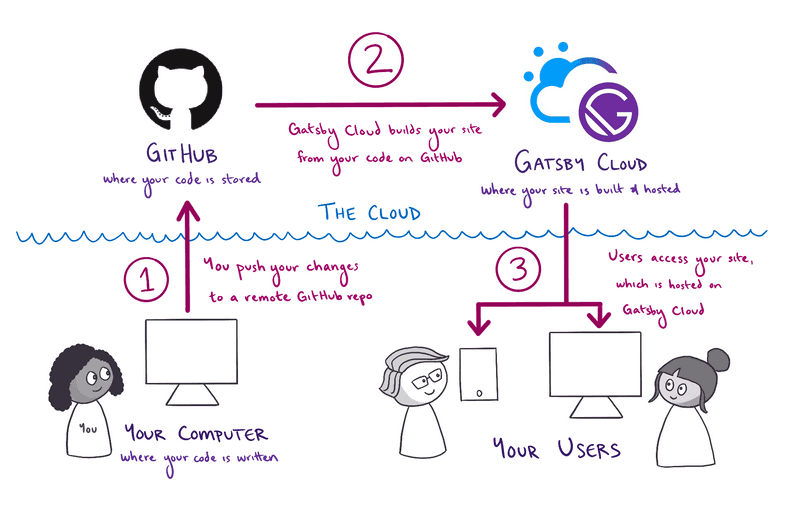
You did it! Your Gatsby site is now online! 👏

**Every time you push a new change to the main branch of your GitHub repo, Gatsby Cloud will see the changes and automatically start a build for the new version of your site.**

**Tip:** There will be a unique URL for each build (like https://build-49535320-b3ae-4761-bbeb-f8f7fa07e0fc.gtsb.io/), and a URL that always has the latest build (like my-first-gatsby-site-main.gatsbyjs.io). You’ll mostly want to share the human-readable URL, so that people can always see the most up-to-date version of your site. But in some cases (like when you’re trying to debug a specific build of your site) it can be helpful to use the unique build URL.

**Summary**

In this section, you learned how to create a new Gatsby site and deploy it online using Gatsby Cloud. As a quick review, here’s the diagram outlining the process:

[](https://www.gatsbyjs.com/static/0fd27b745c1de708f034eaf97c4416e0/d61c2/deployment-workflow.png)

*Expand for detailed description*

**Key takeaways**

* To create a new Gatsby site from the command line, you can run the gatsby new command.
* To run your site locally, use the gatsby develop command. You can view your site in a web browser at localhost:8000.
* Gatsby Cloud is an infrastructure platform specifically optimized for building, deploying, and hosting Gatsby sites.
  + When you push a new commit to the main branch of the GitHub repository for your site, Gatsby Cloud will detect the changes, rebuild a new version of your site, and then redeploy it.

**Share Your Feedback!**

Our goal is for this Tutorial to be helpful and easy to follow. We’d love to hear your feedback about what you liked or didn’t like about this part of the Tutorial.

Use the “Was this doc helpful to you?” form at the bottom of this page to let us know what worked well and what we can improve.

**What’s coming next?**

Now that you have a default Gatsby site up and running, it’s time to make it your own. In Part 2 of the Tutorial, you’ll learn how to use React to customize the design and contents of your site.

[Continue to Part 2](https://www.gatsbyjs.com/docs/tutorial/getting-started/part-2/)

**Part 2: Use and Style React Components**

**TABLE OF CONTENTS**

* [Introduction](https://www.gatsbyjs.com/docs/tutorial/getting-started/part-2/#introduction)
* [A quick intro to React](https://www.gatsbyjs.com/docs/tutorial/getting-started/part-2/#a-quick-intro-to-react)
  + [What is React?](https://www.gatsbyjs.com/docs/tutorial/getting-started/part-2/#what-is-react)
  + [What is a React component?](https://www.gatsbyjs.com/docs/tutorial/getting-started/part-2/#what-is-a-react-component)
* [Create a page component](https://www.gatsbyjs.com/docs/tutorial/getting-started/part-2/#create-a-page-component)
  + [Task: Update the home page content](https://www.gatsbyjs.com/docs/tutorial/getting-started/part-2/#task-update-the-home-page-content)
  + [Task: Create a new page component for an About page](https://www.gatsbyjs.com/docs/tutorial/getting-started/part-2/#task-create-a-new-page-component-for-an-about-page)
* [Use the <Link> component](https://www.gatsbyjs.com/docs/tutorial/getting-started/part-2/#use-the-link-component)
* [Create a reusable layout component](https://www.gatsbyjs.com/docs/tutorial/getting-started/part-2/#create-a-reusable-layout-component)
* [Style components with CSS Modules](https://www.gatsbyjs.com/docs/tutorial/getting-started/part-2/#style-components-with-css-modules)
* [Summary](https://www.gatsbyjs.com/docs/tutorial/getting-started/part-2/#summary)
  + [Key takeaways](https://www.gatsbyjs.com/docs/tutorial/getting-started/part-2/#key-takeaways)
  + [What's coming next?](https://www.gatsbyjs.com/docs/tutorial/getting-started/part-2/#whats-coming-next)

**Introduction**

In the previous part of the Tutorial, you started your first Gatsby site and used Gatsby Cloud to deploy it to the internet. Now that you have everything all set up, it’s time to make this site your own!

To build out the basic page structure for your blog site, you’ll need to know about React components and how Gatsby uses them.

By the end of this part of the Tutorial, you will be able to:

* Create **page components** to add new pages to your site.
* Add a title to your site using the **Gatsby Head API**
* Import and use a **pre-built component** from another package.
* Create your own **reusable “building block” component**.
* Use component **props** to change the way a component renders.
* Use the children prop to create a wrapper component.

**Prefer a video?**

If you’d rather follow along with a video, here’s a recording of a livestream that covers all the material for Part 2. **Please note:** At the time of recording the Gatsby Head API didn’t exist yet and thus the video contents and text contents are different. Please always follow the written instructions. We’ll do our best to record a new version in the future, thanks for understanding!

**Note**: Parts of this recording may be slightly outdated, but the concepts are generally applicable. For the most up-to-date information, follow along with the written tutorial.

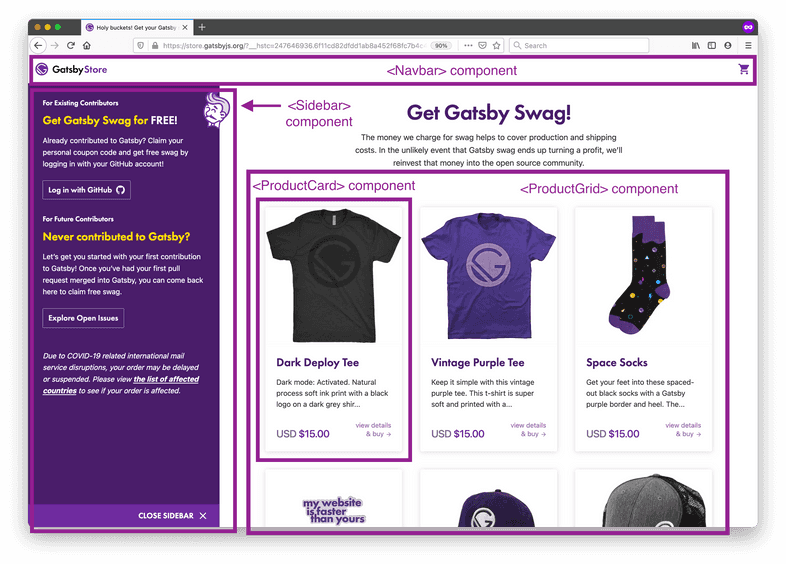
Don’t want to miss any future livestreams? Follow our [Gatsby Twitch channel](https://www.twitch.tv/gatsbyjs).

**A quick intro to React**

**What is React?**

React is the JavaScript library that Gatsby uses under the hood to create user interfaces (UIs). With React, you can break down your UI into smaller, reusable pieces called **components**.

For example, imagine the UI for an online store’s Products page:

[](https://www.gatsbyjs.com/static/e956cd45d790934484f0d93643c87cc7/e0449/ui-built-from-components.png)

To build this page in React, you might have a <Navbar> component for the navigation menu, a <Sidebar> component for extra information displayed to the side of the main content, and a <ProductGrid> component to display all of the products for sale.

You can also create components from other components. For example, you might decide to break down the <ProductGrid> component into a list of multiple <ProductCard> components, which each display the details about a single product. This pattern is called **composition**, since your larger <ProductGrid> component is *composed* of smaller <ProductCard> components.

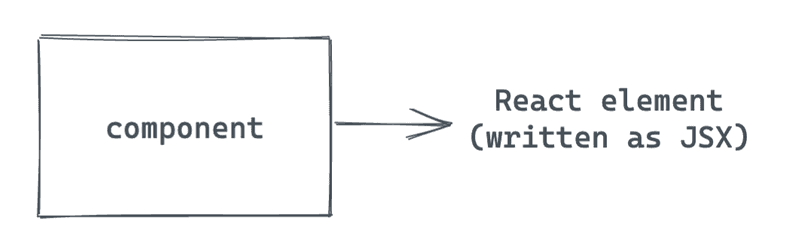
**Try it!**

Look back at the [finished example blog site](https://gatsbytutorialexample.gatsbyjs.io/). How might you break down the pages into components?

Not sure where to start? Look for parts of the UI that repeat within a page or across multiple pages.

**What is a React component?**

Under the hood, a **React component** is a function that returns a React element. A **React element** is an object that React uses to render DOM elements.

[](https://www.gatsbyjs.com/static/d2e8dcd3ec0c8c63d261cb0f9132aa5a/00172/component-without-props.png)

The simplest way to write React elements is with **JSX**. JSX is a JavaScript syntax extension that describes the DOM structure for your component. It looks a bit like having HTML in your JavaScript files:

Copycopy code to clipboard

const hello = <h1>Hello world!</h1>

So a simple React component might look something like this:

Copycopy code to clipboard

const Greeting = () => {

return <h1>Hello world!</h1>

}

You’ll learn more about how to create React components in the next section.

**Create a page component**

There are two main types of components in a Gatsby site. The first type you’ll create are **page components**. A page component contains all the UI elements for a specific page of your site.

In this section, you’ll create two new page components: one for the Home page and one for an About page.

(If you’re new to writing React components, expand the content below for a quick overview.)

**Key React Concept: Writing a React component**

There are three main steps to writing a React component:

1. **Import React** from the 'react' package, so that you can use JSX inside your .js file.
2. **Define your component.** It should be a function that returns a JSX element.
3. **Export your component**, so that it can be used by other parts of your site.

Here’s an example of what that basic structure looks like in code:

my-component.js

Copymy-component.js: copy code to clipboard

// Step 1: Import React. This lets you use JSX inside your .js file.

import \* as React from 'react'

/\* Step 2: Define your component. Note that your

component name should start with a capital letter. \*/

const MyComponent = () => {

return (

<h1>Hi, welcome to my site!</h1>

)

}

/\* Step 3: Export your component so it

can be used by other parts of your app. \*/

export default MyComponent

Your component must return a single React element, but you can put as many elements inside that top-level element as you want. The code snippet below shows an example of a valid component and an invalid component:

Copycopy code to clipboard

import \* as React from 'react'

const ValidComponent = () => {

return (

<div>

<h1>A valid component!</h1>

<p>This will work fine.</p>

<p>

Since there is only one top-level element: the div.

</p>

</div>

)

}

const InvalidComponent = () => {

return (

<h1>This won't work.</h1>

<p>Because there are two elements at the top level.</p>

)

}

If you try to build your site with the code above, you’ll get an error for <InvalidComponent> like this:

Copycopy code to clipboard

Parsing error: Adjacent JSX elements must be

wrapped in an enclosing tag. Did you want a JSX fragment

<>...</>?

**Task: Update the home page content**

Now that you’ve gotten a high-level introduction to React, it’s time to try your hands at writing some React components. To start, you’ll update the content for the home page.

If you haven’t already, open your Gatsby site in Visual Studio Code, and start up your local development server in the command line:

1. Open your command line application.
2. Change directories into the folder for your Gatsby site.
3. Run gatsby develop.
4. Open localhost:8000 in your web browser.
5. Open your src/pages/index.js file. Replace its contents with the following. (Notice how the component’s structure matches the three steps for writing React components?)

src/pages/index.js

Copysrc/pages/index.js: copy code to clipboard

// Step 1: Import React

import \* as React from 'react'

// Step 2: Define your component

const IndexPage = () => {

return (

<main>

<h1>Welcome to my Gatsby site!</h1>

<p>I'm making this by following the Gatsby Tutorial.</p>

</main>

)

}

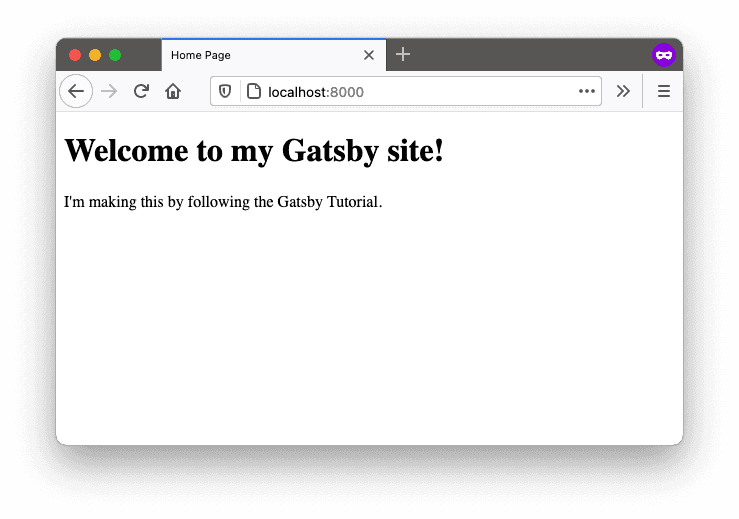
// You'll learn about this in the next task, just copy it for now

export const Head = () => <title>Home Page</title>

// Step 3: Export your component

export default IndexPage

1. Go to localhost:8000 in a web browser. (You might need to wait a moment while your development server rebuilds.) Once your page updates, it should look something like this:

[](https://www.gatsbyjs.com/static/d68e4f183db77b362698a9ec74a1951f/f1d1f/index-page.png)

**Key Gatsby Concept** 💡

Gatsby automatically creates pages for React components that are the default export of files in the src/pages directory.

If a user tries to visit the URL for a page that doesn’t actually exist, Gatsby will use the src/pages/404.js page component to display an error instead. Go ahead and give it a try! (If you’re trying it on localhost:8000 you’ll need to click the “Preview custom 404 page” button on the development 404 page.)

**Task: Create a new page component for an About page**

Now that you’ve updated the existing Home page, try creating a new page from scratch. Make an About page, so that you can tell people a little about yourself.

1. Create a new file: src/pages/about.js. Use the code below as a starting point for your About page. (Feel free to switch up the content to make it more specific to you. Maybe share your favorite food or go-to vacation spot!)

src/pages/about.js

Copysrc/pages/about.js: copy code to clipboard

// Step 1: Import React

import \* as React from 'react'

// Step 2: Define your component

const AboutPage = () => {

return (

<main>

<h1>About Me</h1>

<p>Hi there! I'm the proud creator of this site, which I built with Gatsby.</p>

</main>

)

}

// Step 3: Export your component

export default AboutPage

1. Add a page title to your page. Gatsby lets you define a <title> and other [document metadata](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/head) with the [Gatsby Head API](https://www.gatsbyjs.com/docs/reference/built-in-components/gatsby-head/). You have to export a component called Head from your page template to apply the metadata. Adding such metadata helps search engines like Google to better understand your site. For this tutorial you’ll only be adding titles to pages but you can also later add other metadata.

src/pages/about.js

Copysrc/pages/about.js: copy code to clipboard

import \* as React from 'react'

const AboutPage = () => {

return (

<main>

<h1>About Me</h1>

<p>Hi there! I'm the proud creator of this site, which I built with Gatsby.</p>

</main>

)

}

export const Head = () => <title>About Me</title>

export default AboutPage

**Key Gatsby Concept** 💡

You can use the [Gatsby Head API](https://www.gatsbyjs.com/docs/reference/built-in-components/gatsby-head/) by exporting a named function called Head in your pages and page templates (e.g. the ones used by createPage or the File System Route API).

Be sure to capitalize Head and please note that exporting this named function inside a component like Layout won’t add the metadata to the <head>. The above works because you’re exporting Head in a page inside src/pages.

You can add any valid <head> tags inside the function and they’ll be added to the page, for example:

Copycopy code to clipboard

export const Head = () => (

<>

<title>About Me</title>

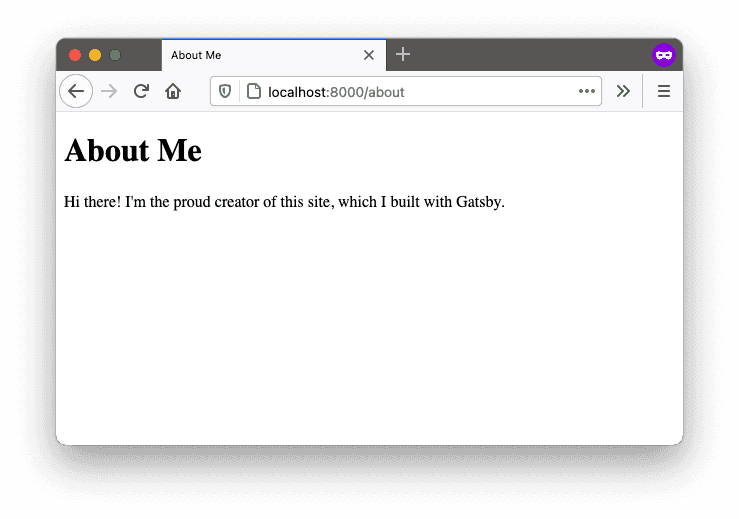
<meta name="description" content="Your description" />

</>

)

After going through this tutorial, be sure to check out [Adding an SEO Component](https://www.gatsbyjs.com/docs/how-to/adding-common-features/adding-seo-component/).

1. In a web browser, visit localhost:8000/about. When your development server finishes rebuilding your site, the About page should look something like this:

[](https://www.gatsbyjs.com/static/51a6cf2b743248bf809025d7e6b4b3ba/f1d1f/about-page.png)

**Key Gatsby Concept** 💡

Pages created in the src/pages directory use the name of the file as the route for the page.

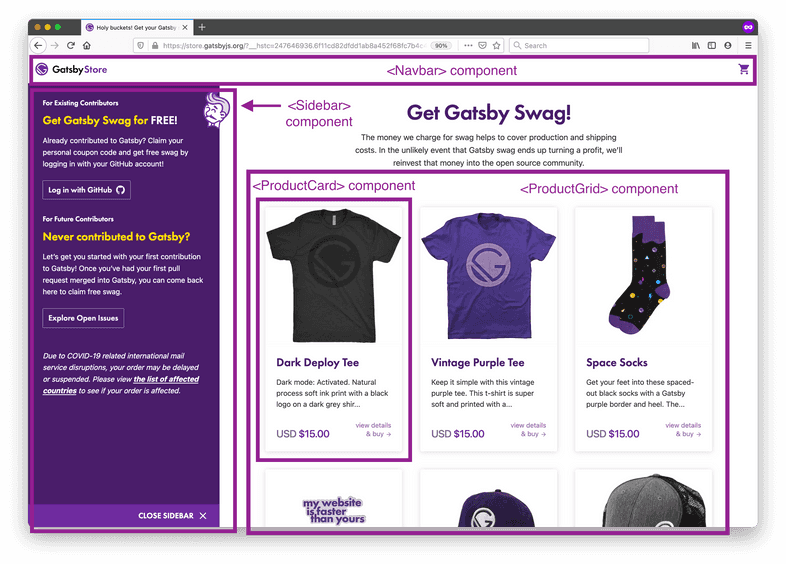
For example, if you had a file called src/pages/garden-gnomes.js, you could access that page at localhost:8000/garden-gnomes.

**Use the**<Link>**component**

Now that you’ve built a few page components, it’s time to look at the other type of React components in a Gatsby site: **“building-block” components**.

**Note:** The term building-block component isn’t an official technical term. It’s just the best name we could come up with to describe this kind of component.

Building-block components are smaller components that represent just a part of a page’s user interface (instead of an entire page). Think back to the store website example from the [“What is React?”](https://www.gatsbyjs.com/docs/tutorial/getting-started/part-2/#what-is-react) section. The Navbar, Sidebar, ProductGrid, and ProductCard components are examples of building-block components. You can combine several smaller building-block components into a larger page component.

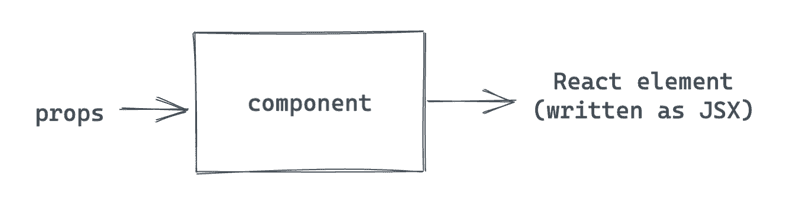
[](https://www.gatsbyjs.com/static/e956cd45d790934484f0d93643c87cc7/e0449/ui-built-from-components.png)

One of the powerful things about building-block components is that you can reuse the same component in multiple places across your site. This is especially useful for pieces of your UI that share a similar structure but render different values.

For example, each of the ProductCard components in the product grid above shows a photo of the product, the product name, a short description, the price, and a link to the product page. The exact values change for each product, but the general structure stays the same. By making the ProductCard component dynamic, you can reuse the same code for all the products in the grid!

React has a built-in feature to help you make your components dynamic: **properties** (or **props**, for short).

**Key React Concept: Components with props**

[](https://www.gatsbyjs.com/static/c058cc46416d6b9d5d27e571c885c936/6569d/component-with-props.png)

So far, your blog site has two separate pages (Home and About), but the only way to get from one page to the other is to update the URL manually. It would be nice to add links to make it easier to switch between pages on your site.

The Link component is an example of a **pre-built** component that you can use in your site. In other words, the Link component is defined and maintained by another package (in this case, the Gatsby package). That means you can import it and use it in your own components without knowing too much about how it works under the hood.

The Link component lets you add a link to another page in your Gatsby site. It’s similar to an HTML <a> tag, but with some extra performance benefits. The Link component takes a prop called to, which is similar to the <a> tag’s href attribute. The value should be the URL path to the page on your site you want to link to.

**Key Gatsby Concept** 💡

The Gatsby Link component provides a performance feature called **preloading**. This means that the resources for the linked page are requested when the link scrolls into view or when the mouse hovers on it. That way, when the user actually clicks on the link, the new page can load super quickly.

Use the Link component for linking between pages within your site. For external links to pages not created by your Gatsby site, use the regular HTML <a> tag.

Follow the steps below to add Link components to your Home and About pages.

1. On the Home page, import the Link component from the Gatsby package and add a link to your About page.

src/pages/index.js

Copysrc/pages/index.js: copy code to clipboard

import \* as React from 'react'

import { Link } from 'gatsby'

const IndexPage = () => {

return (

<main>

<h1>Welcome to my Gatsby site!</h1>

<Link to="/about">About</Link>

<p>I'm making this by following the Gatsby Tutorial.</p>

</main>

)

}

export const Head = () => <title>Home Page</title>

export default IndexPage

1. On the About page, import the Link component from the Gatsby package and add a link to your Home page.

src/pages/about.js

Copysrc/pages/about.js: copy code to clipboard

import \* as React from 'react'

import { Link } from 'gatsby'

const AboutPage = () => {

return (

<main>

<h1>About Me</h1>

<Link to="/">Back to Home</Link>

<p>Hi there! I'm the proud creator of this site, which I built with Gatsby.</p>

</main>

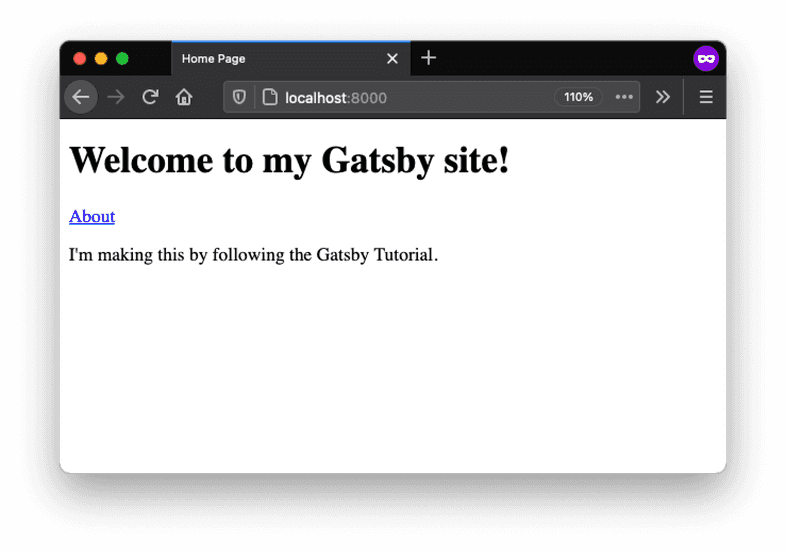
)

}

export const Head = () => <title>About Me</title>

export default AboutPage

1. In your web browser, click on each of the links to make sure they’re working correctly.

[](https://www.gatsbyjs.com/static/ae8f5343969dc89888aab0e675454030/07d7d/index-page-with-link.png)

[](https://www.gatsbyjs.com/static/101233741dc1abc9cef2c79533e0c7ea/07d7d/about-page-with-link.png)

**Create a reusable layout component**

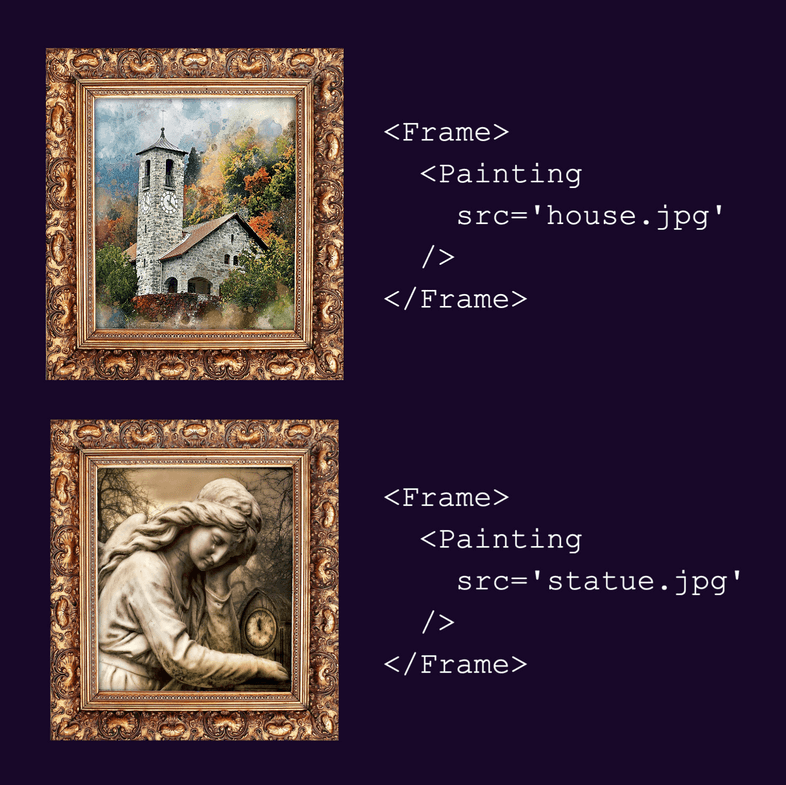
If you take another look at the [finished example blog](https://gatsbytutorialexample.gatsbyjs.io/), you might notice that there are some repeated parts of the UI across each page, like the site title and the navigation menu.

You could copy those elements into each page of your site separately. But imagine your site had dozens (or even thousands) of pages. If you wanted to make a change to the structure of your navigation menu, you’d have to go and update every one of those files separately. Yuck.

Instead, it would be better to create one common Layout component that groups all the shared elements to reuse across multiple pages. That way, when you need to make updates to the layout, you can make the change in one place and it will automatically be applied to all the pages using that component.

In this section, you’ll create your first **custom** building-block component: Layout. To do that, you’ll need to use a special React prop called children.

**Key React Concept: Components with children**

[](https://www.gatsbyjs.com/static/57158e9ba90e928514d13add8baa85e9/36eca/children-example.png)

Follow the steps below to create a Layout component and add it to your Home and About pages.

1. Create a new file called src/components/layout.js. Insert the following code to define your Layout component. This component will render a dynamic heading (from the pageTitle prop), a list of navigation links, and the contents passed in with the children prop. To improve accessibility, there’s also a <main> element wrapping the page-specific elements (the <h1> heading and the contents from children).

src/components/layout.js

Copysrc/components/layout.js: copy code to clipboard

import \* as React from 'react'

import { Link } from 'gatsby'

const Layout = ({ pageTitle, children }) => {

return (

<div>

<nav>

<ul>

<li><Link to="/">Home</Link></li>

<li><Link to="/about">About</Link></li>

</ul>

</nav>

<main>

<h1>{pageTitle}</h1>

{children}

</main>

</div>

)

}

export default Layout

**Syntax Hint**: You might have noticed that the Layout component uses a slightly different syntax for its props.

Now instead of looking like this:

Copycopy code to clipboard

const Layout = (props) => {

...

}

…it looks like this:

Copycopy code to clipboard

const Layout = ({ pageTitle, children }) => {

...

}

This is a JavaScript technique called **destructuring**. It’s basically a shortcut for defining variables based on an object’s properties. It’s like saying, “Take the object that gets passed into this function, and unpack its pageTitle and children properties into their own variables.”

In other words, it’s a shorter way to do the following:

Copycopy code to clipboard

const Layout = (props) => {

const pageTitle = props.pageTitle

const children = props.children

...

}

1. Update your Home page component to use the Layout component instead of the hard-coded Link component you added in the previous section.

src/pages/index.js

Copysrc/pages/index.js: copy code to clipboard

import \* as React from 'react'

import Layout from '../components/layout'

const IndexPage = () => {

return (

<Layout pageTitle="Home Page">

<p>I'm making this by following the Gatsby Tutorial.</p>

</Layout>

)

}

export const Head = () => <title>Home Page</title>

export default IndexPage

1. Update your About page component to use the Layout component as well.

src/pages/about.js

Copysrc/pages/about.js: copy code to clipboard

import \* as React from 'react'

import Layout from '../components/layout'

const AboutPage = () => {

return (

<Layout pageTitle="About Me">

<p>Hi there! I'm the proud creator of this site, which I built with Gatsby.</p>

</Layout>

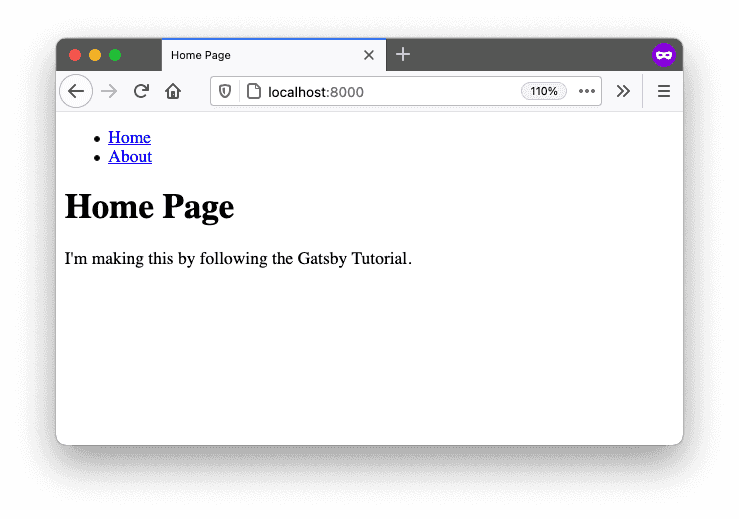
)

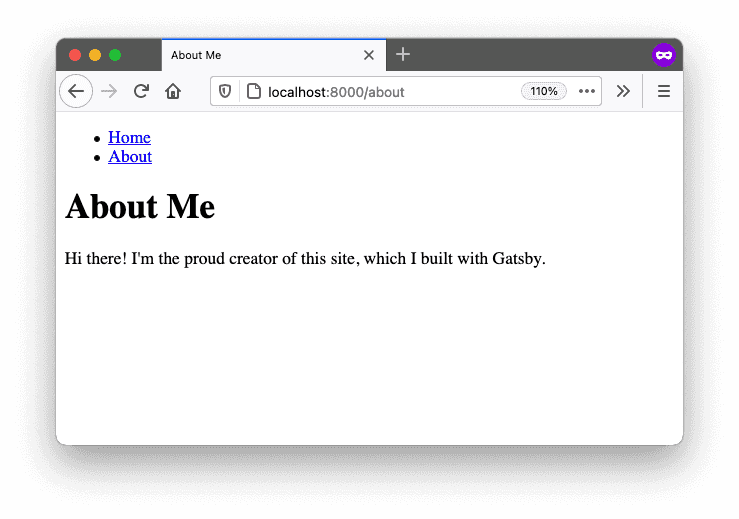
}

export const Head = () => <title>About Me</title>

export default AboutPage

1. Check your Home and About pages in a web browser to make sure your new Layout component is working:

[](https://www.gatsbyjs.com/static/2bb409ab1579861c6d21fbcefc8aa216/f1d1f/index-page-with-layout.png)

[](https://www.gatsbyjs.com/static/6afc66d0050f0a65bc2018f46e2fc11d/f1d1f/about-page-with-layout.png)

**Style components with CSS Modules**

Now that you’ve got your page structure set up, it’s time to add some style and make it cute!

Gatsby isn’t strict about what styling approach you use. You can pick whatever system you’re most comfortable with.

In this Tutorial, you’ll use **CSS Modules** to style your components. This means that styles will be scoped to components, which helps avoid class naming collisions between components. Gatsby is automatically configured to handle CSS Modules - no extra setup necessary!

**Key Styling Concept: CSS Modules**

Follow the steps below to style your Layout component using CSS Modules.

1. Create a new file: src/components/layout.module.css. (The .module.css part at the end is important! That’s what tells Gatsby that these styles are using CSS Modules.)
2. Start by adding a single .container class:

src/components/layout.module.css

Copysrc/components/layout.module.css: copy code to clipboard

.container {

margin: auto;

max-width: 500px;

font-family: sans-serif;

}

1. Then import that class into your Layout component .js file, and use the className prop to assign it to the top-level <div> element:

src/components/layout.js

Copysrc/components/layout.js: copy code to clipboard

import \* as React from 'react'

import { Link } from 'gatsby'

import { container } from './layout.module.css'

const Layout = ({ pageTitle, children }) => {

return (

<div className={container}>

<nav>

<ul>

<li><Link to="/">Home</Link></li>

<li><Link to="/about">About</Link></li>

</ul>

</nav>

<main>

<h1>{pageTitle}</h1>

{children}

</main>

</div>

)

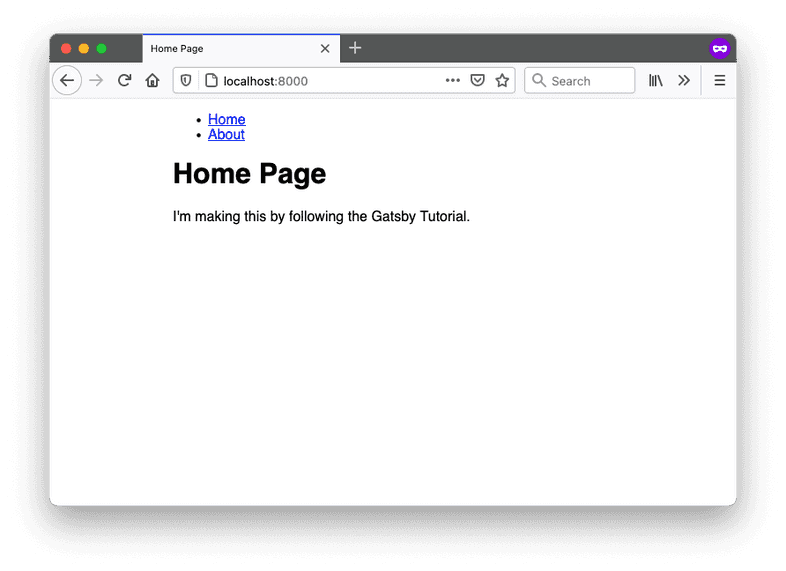
}

export default Layout

**Syntax Hint:** To apply classes to React components, use the className prop. (This is another example of a built-in prop that React automatically knows how to handle.)

This might be confusing if you’re used to using the class attribute on HTML elements. Do your best to not mix them up!

1. When you open your local site in a web browser, you should now see the font has changed and the content is more centered on the page.

[](https://www.gatsbyjs.com/static/0c3679a8b3c806a5bb82b41799b680b6/1568e/index-page-container-styled.png)

1. Now that you’ve seen how to style a single element for your component, add some more styles to apply to the other elements in your Layout component.

src/components/layout.module.css

Copysrc/components/layout.module.css: copy code to clipboard

.container {

margin: auto;

max-width: 500px;

font-family: sans-serif;

}

.heading {

color: rebeccapurple;

}

.nav-links {

display: flex;

list-style: none;

padding-left: 0;

}

.nav-link-item {

padding-right: 2rem;

}

.nav-link-text {

color: black;

}

1. Import the new classes into your Layout component, and apply each class to the corresponding element.

src/components/layout.js

Copysrc/components/layout.js: copy code to clipboard

import \* as React from 'react'

import { Link } from 'gatsby'

import {

container,

heading,

navLinks,

navLinkItem,

navLinkText

} from './layout.module.css'

const Layout = ({ pageTitle, children }) => {

return (

<div className={container}>

<nav>

<ul className={navLinks}>

<li className={navLinkItem}>

<Link to="/" className={navLinkText}>

Home

</Link>

</li>

<li className={navLinkItem}>

<Link to="/about" className={navLinkText}>

About

</Link>

</li>

</ul>

</nav>

<main>

<h1 className={heading}>{pageTitle}</h1>

{children}

</main>

</div>

)

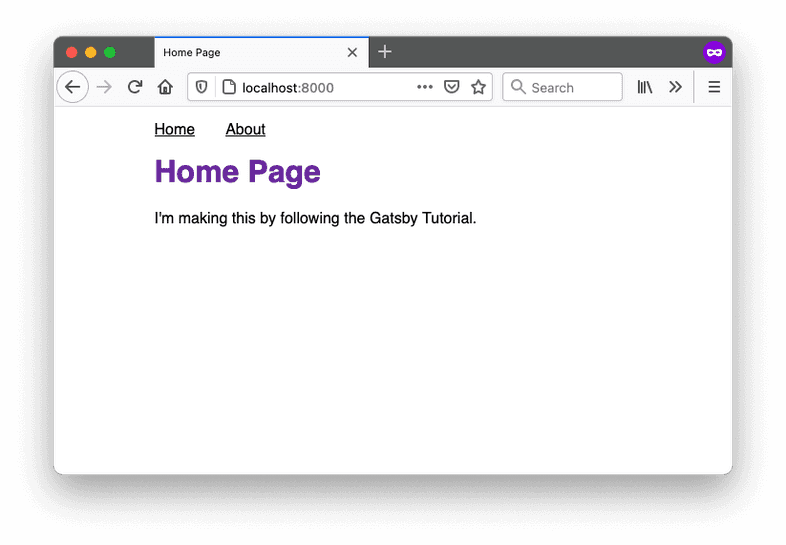
}

export default Layout

**Syntax Hint:** In CSS, the convention is to name classes using kebab case (like .nav-links). But in JavaScript, the convention is to name variables using camel case (like navLinks).

Luckily, when you use CSS Modules with Gatsby, you can have both! Your kebab-case class names in your .module.css files will automatically be converted to camel-case variables that you can import in your .js files.

1. Once your development server finishes rebuilding your site, you should see your new styles applied in your web browser:

[](https://www.gatsbyjs.com/static/60df3db72ca03c1cf5fe715c4ccd82de/31aff/index-page-styled.png)

**Want to see how it all fits together?** Check out the finished state of the [GitHub repo for the example site](https://github.com/gatsbyjs/tutorial-example).

**Summary**

Congratulations, you’ve made it to the end of Part 2! 🥳 (That was a long one!)

Take a moment to think back on what you’ve learned so far. Challenge yourself to answer the following questions from memory:

* What’s the difference between a page component and a building-block component?
* How do you add a new page to your Gatsby site?
* How do you add a title to a page?
* What are the three steps for writing a new React component?
* What are props and when might you use them?
* What is the children prop and why is it useful?

**Ship It!** 🚀

Before you move on, deploy your changes to your live site on Gatsby Cloud so that you can share your progress!

First, run the following commands in a terminal to push your changes to your GitHub repository. (Make sure you’re in the top-level directory for your Gatsby site!)

Copycopy code to clipboard

git add .

git commit -m "Finished Gatsby Tutorial Part 2"

git push

Once your changes have been pushed to GitHub, Gatsby Cloud should notice the update and rebuild and deploy the latest version of your site. (It may take a few minutes for your changes to be reflected on the live site. Watch your build’s progress from your [Gatsby Cloud dashboard](https://www.gatsbyjs.com/dashboard/).)

**Key takeaways**

* React is a library that helps you break down your UI into smaller pieces called components. A component is a function that returns a React element. React elements can be written in JSX.
* **Page components** contain all the UI elements for a specific page of your site. Gatsby automatically creates pages for components that are the default exports of files in the src/pages directory. The name of the file will be used as the route for the page.
* You can use the **Gatsby Head API** by exporting a named function Head to define metadata for the page.
* **Building-block components** are smaller reusable parts of your UI. They can be imported into page components or other building block components.
* You can import **pre-built** components (like Link) from other packages, or you can write your own **custom** components from scratch (like Layout).
* You can use **props** to change how a component renders. You can define your own props when you build a component. React also has some built-in props, like children and className.
* Gatsby isn’t opinionated about what styling approach you want to use, but it works with **CSS Modules** by default.

**Share Your Feedback!**

Our goal is for this Tutorial to be helpful and easy to follow. We’d love to hear your feedback about what you liked or didn’t like about this part of the Tutorial.

Use the “Was this doc helpful to you?” form at the bottom of this page to let us know what worked well and what we can improve.

**What’s coming next?**

In Part 3 of the Tutorial, you’ll learn about how to use Gatsby plugins to add more pre-built functionality to your site.

[Continue to Part 3](https://www.gatsbyjs.com/docs/tutorial/getting-started/part-3/)

**[Start building today on Gatsby Cloud!](https://www.gatsbyjs.com/dashboard/signup)**

[Edit this page on GitHub](https://github.com/gatsbyjs/gatsby/blob/master/docs/docs/tutorial/getting-started/part-2/index.mdx)