Welcome!

You don't have to be an expert in a topic to write about it -- this entire website is open source, so even if you make a mistake, another contributor will help you correct it before the PR gets merged.

Before you begin writing, make sure to read the rest of this style guide.

What kinds of docs can I write?

Please see the Docs Structure for an overview of the different types of documents, as well as guidelines for how to format them.

Writing process

Think of your audience

Before you begin writing, answer these questions. Sample answers have been included:

Question: Who will read my writing? *Answer: Developers with knowledge and experience coding in HTML, CSS, and JS but not necessarily React or GraphQL.*

Question: What do I hope my readers will know and/or be able to do after reading it? *Answer (example): I hope my readers will be able to successfully add search to their Gatsby site.*

Once you answer those questions, create an outline of the topic and think about any coding examples you'll use (if applicable). This helps to organize your thoughts and make the writing process easier.

Research

Many times, the information that needs to go in your document already exists somewhere.

Avoid copying and pasting huge chunks of other people's work. Instead, use their work to learn so you can write your own document. If you do quote someone's work verbatim, reference where you got the information.

If the content is already somewhere else on the site, feel free to copy and paste without quoting or referencing.

Possible sources of great research materials:

- blogposts (on gatsbyjs.com and other sites)
- docs (on gatsbyjs.com and other sites)
- video tutorials
- Discord or Twitter conversations
- search engine results
- presentations you or others have given
- textbooks
- dreams
- anything else you can think of

Write drafts and get feedback

Technical writing, or the literature of science and technology, is difficult because it requires you to take a technical (usually abstract) topic and explain it in a clear, accurate, and objective manner. You'll likely go through several rounds of proofreading and editing before you're happy with your writing.

Also, there's a community of contributors to support you. Bounce ideas off of them and ask for input on your writing in the Gatsby Discord and in the GitHub repo.

Word choice

Use "you" as the pronoun

In English, your articles should use the second person ("you") to provide a conversational tone. This way, the text and instructions seem to speak directly to the person reading it. Try to avoid using the first person ("I", "we", "let's", and "us").

For other languages, refer to each translation's guidelines (if applicable) for consistent phrasing. When appropriate, we suggest starting with the informal "you" to keep a conversational tone.

Using "you" in English is also more accurate than saying "we," because typically only one person is reading the tutorial or guide at a time and the person who wrote the tutorial is not actually going through it with them, so "we" would be inaccurate. You might notice that some technical documentation uses third-person pronouns and nouns like "they" and "the user," which add more distance and feel colder than the conversational and warm "you" and "your."

When updating a doc to adhere to this part of the Gatsby Style Guide, one exception in English is when "we" refers to Gatsby's core processes. The subject is the code in this case, rather than a teacher/reader connotation, and should be rewritten or restructured to not confuse the reader about what they are responsible for doing when something is happening automatically behind the scenes.

Avoid "easy" and "simple"

Avoid using words like "easy", "simple" and "basic" because if users have a hard time completing the task that is supposedly "easy," they will question their abilities. Consider using more specific descriptors; for example, when you say the phrase "deployment is easy," what do you really mean? Is it easy because it takes fewer steps than another option? If so, use the most specific descriptor possible, which in that case would be "this deployment method involves fewer steps than other options."

For even more inclusive docs, avoid phrases that assume a reader's experience or skill level, like "just deploy it and you're done" or "for a refresher (referring to a completely different doc that someone may not have read)". Often, rephrasing results in stronger sentences that appeal to a wider range of contexts.

Avoid emojis, slang, and metaphors

Avoid using emojis or emoticons in the Docs and idiomatic expressions / slang, or metaphors. Gatsby has a global community, and the cultural meaning of an emoji, emoticon, or slang may be different around the world. Use your best judgment! Also, emojis can render differently on different systems.

Define jargon

Articles should be written with short, clear sentences, and use as little jargon as necessary.

> *Jargon: (n.) special words or expressions that are used by a particular profession or group and are difficult for others to understand: legal jargon.*

All jargon should be defined immediately in plain English. In other words, pretend like your readers have basic coding experience but not necessarily experience with PWAs and the JAMstack (see what happened there? I just used two jargon words that need to be defined); you need to define words that newcomers might have a hard time understanding.

Writing style

Write concisely

Concise writing communicates the bare minimum without redundancy. Strive to make your writing as short as possible; this practice will often lead to more accurate and specific writing.

Use clear hyperlinks

Hyperlinks should contain the clearest words to indicate where the link will lead you. The use of the title attribute on hyperlinks should be avoided for accessibility reasons.

```
<!-- Good -->

[Gatsby Cloud](https://www.gatsbyjs.com/cloud/)

<!-- Bad -->

[here](https://www.gatsbyjs.com/cloud/ "Gatsby Cloud")
```

In tutorials that are meant for beginners, use as few hyperlinks as possible to minimize distractions. In docs, it's ok to include as many hyperlinks as necessary to provide relevant and interesting information and resources.

Use relative hyperlinks for local links

When referencing another page within gatsbyjs.com hyperlinks should use relative paths (not include the full domain). This guarantees that all links function when running locally or in preview.

```
<!-- Good -->

[Gatsby's glossary](/docs/glossary/)

<!-- Bad -->

[Gatsby's glossary](https://www.gatsbyjs.com/docs/glossary/)
```

Mark localhost URLs as code strings

Unless you're running `gatsby develop` or `gatsby build` locally, localhost links will not work. Therefore it's recommended to list these URL references as code blocks so there aren't invalid links throughout the docs.

```
<!-- Good -->

open your site with `http://localhost:8000/`

<!-- Bad -->

open your site with [http://localhost:8000/](http://localhost:8000/)
```

Indicate when something is optional

When a paragraph or sentence offers an optional path, the beginning of the first sentence should indicate that it's optional. For example, "if you'd like to learn more about xyz, see our reference guide" is clearer than "Go to the

reference guide if you'd like to learn more about xyz."

This method allows people who would *not* like to learn more about xyz to stop reading the sentence as early as possible. This method also allows people who *would* like to learn more about xyz to recognize the opportunity to learn quicker instead of accidentally skipping over the paragraph.

Abbreviate terms

If you want to abbreviate a term in your article, write it out fully first, then put the abbreviation in parentheses. After that, you may use the abbreviation going for the rest of the article. For example, "In computer science, an abstract syntax tree (AST) is ..."

Use SEO optimized titles

This explains how to account for Search Engine Optimization (SEO) and create a doc that shows up in search engines like Google or Bing.

When you create the new guide or tutorial under `/docs/`, you'll either create a file or a folder if there will be images pulled into the doc.

File: `querying-data-with-graphql.md`

or

Folder: `querying-data-with-graphql` `querying-data-with-graphql/index.md` `querying-data-with-graphql/graphql-image.png`

The `.md` title or the folder title gets turned into the URL route automatically.

Article titles should be short and reflect the main theme of the article to help readers quickly find relevant info. Many people use a search engine to find topics like "gatsby GraphQL", so the article title should ideally reflect common search terms.

Here are some title examples:

- Creating & Modifying Pages
- Adding a 404 Page
- Querying Data with GraphQL

The folder name is used in the URL, so only use dashes -, numbers 0-9, and lowercase letters a-z for it.

Here are some folder name examples:

- creating-and-modifying-pages
- adding-a-404-page
- querying-data-with-graphql

Note: Just to clarify, you can include special characters in the article title but *not* in the `.md` file name or folder name (e.g. title: What is GraphQL? and Folder Name: what-is-graphql).

Grammar and formatting

Format titles and headers

Title case article titles (each major word is uppercase). Sentence case article headings (only the initial word is uppercase). Neither need punctuation at the end of the phrase unless a question mark is required. Article titles do

not take the Oxford comma and use the ampersand in place of “and.” Article headings do take the Oxford comma and use the word “and.”

Titles are automatically formatted as h1. Mark up article headings as h2 and subheads as h3 or h4 as needed. Most article headings are conceptually and rhetorically at the same level as each other; avoid unnecessary complexity and mark them up as h2 unless they’re true subheads.

Article title or document title:

> *Salty, Sweet & Spicy*

Article header or subhead:

> *Salty, sweet, and spicy*

Titles should aim to be brief while still conveying a comprehensive meaning of the article; headings have more leeway in terms of length. Because titles show up throughout the docs in navigation elements (like breadcrumbs, and sidebar navigation) there is a preference for shorter names to help mitigate visual clutter.

Format code blocks, inline code, and images

Use the following as reference when creating and editing docs:

- formatting inline code and code blocks
- adding images to articles. If the images aren’t already hosted somewhere else on the web, you’ll need to put them online yourself. A good way to do this is to commit them to a GitHub repository of your own, then push them to GitHub. Then you can right-click the image and copy its image source. And don't forget image alt text for accessibility! For help with crafting efficient screen reader text, refer to the W3C's alt decision tree.
- header formatting. Avoid using H1 header; that is reserved for the title of each document.

Code formatting: Inline code

Ensure that variables, component names, function names, and packages that appear inline are escaped with backticks:

```
<!-- Good -->

The plugin `gatsby-transformer-something` provides several useful options, such as
the `somethingArgs` variable that can be passed in to `createSchemaCustomization`.

<!-- Bad -->

The plugin gatsby-transformer-something provides several useful options, such as the
somethingArgs variable that can be passed in to createSchemaCustomization.
```

Code formatting: Type tab

Each code snippet will include a tab showing the language type the snippet contains. For example, the following YAML snippet will show a "YAML" tab...

````
```yaml
- id: John Smith
  bio: Thinks documentation is the coolest.
````
````

````
  twitter: @j
```
````

...like so:

```
- id: John Smith
  bio: Thinks documentation is the coolest.
  twitter: @j
```

Please use the following language keywords where appropriate:

- `javascript` or `js`
- `jsx`
- `graphql`
- `html`
- `css`
- `shell`
- `yaml`
- `markdown`
- `diff`
- `flow`

If a language keyword is omitted, the type will show as `TEXT` (as shown above).

Code formatting: Titles

Where appropriate, add code titles to your code blocks. Switching between multiple files in the course of the document can confuse some readers. It's best to explicitly tell them where the code example should go. You can use syntax highlighting as usual, then add `:title=your-path-name` to it. Use it like so:

````
```javascript:title=src/util/alert.js
const s = "I solemnly swear that I'm up to no good."
alert(s)
```
````

Which will then look like:

```
const s = "I solemnly swear that I'm up to no good."
alert(s)
```

Code formatting: Line highlighting

You may also choose to include line highlighting in your code snippets, using the following keywords as comments inline in the snippet:

`highlight-line` : highlights the current line

````
```javascript:title=gatsby-config.js
module.exports = {
  siteMetadata: {
```
````
````

````
    title: `GatsbyJS`, // highlight-line
    siteUrl: `https://www.gatsbyjs.com`,
  },
}
```
````

```
module.exports = {
  siteMetadata: {
    title: `GatsbyJS`, // highlight-line
    siteUrl: `https://www.gatsbyjs.com`,
  },
}
```

`highlight-next-line` : highlights the next line

````
```javascript:title=gatsby-config.js
module.exports = {
  siteMetadata: {
    title: `GatsbyJS`,
    // highlight-next-line
    siteUrl: `https://www.gatsbyjs.com`,
  },
}
```
````

```
module.exports = {
  siteMetadata: {
    title: `GatsbyJS`,
    // highlight-next-line
    siteUrl: `https://www.gatsbyjs.com`,
  },
}
```

`highlight-start` & `highlight-end` : highlights a range

````
```javascript:title=gatsby-config.js
module.exports = {
  // highlight-start
  siteMetadata: {
    title: `GatsbyJS`,
    siteUrl: `https://www.gatsbyjs.com`,
  },
  // highlight-end
}
```
````

```
module.exports = {
  // highlight-start
```
```

```
  siteMetadata: {
    title: `GatsbyJS`,
    siteUrl: `https://www.gatsbyjs.com`,
  },
  // highlight-end
}
```

Capitalize proper nouns

Proper nouns should use correct capitalization when possible. Below is a list of words as they should appear in blog posts, docs, and other learning materials on this website.

- Gatsby
- GraphQL
- JavaScript (capital letters in "J" and "S" and no abbreviations)
- TypeScript
- WordPress
- Markdown
- Node.js
- webpack (should always in lower-case letters, except at the beginning of a sentence)

A full-stack developer (adjective form with a dash) works on the full stack (noun form with no dash). The same goes with many other compound terms.

Use frontend for both adjective and noun forms as it's more common and easier to maintain. For example, a frontend developer works on the frontend. The same goes for backend.

End users are spelled out with two words, rather than hyphenating with a dash.

Use active voice

Use active voice instead of passive voice. Generally, it's a more concise and straightforward way to communicate a subject. For example:

- (passive) The for loop in JavaScript is used by programmers to...
- (active) Programmers use the for loop in JavaScript to...

Make lists clear with the Oxford Comma

Use the Oxford Comma except in titles. It is a comma used after the penultimate item in a list of three or more items, before ‘and’ or ‘or’ e.g. an Italian painter, sculptor, and architect. It makes things clearer.

Prefer US English

For words that have multiple spellings, prefer the US English word over British or Canadian English. For example:

- `color` over `colour`
- `behavior` over `behaviour`

Use apps that help you edit

Use the Hemingway App. There’s nothing magical about this tool, but it will automatically detect widely agreed-upon style issues:

- passive voice
- unnecessary adverbs

- words that have more common equivalents

The Hemingway App will assign a “grade level” for your writing. You should aim for a grade level of 6. Another tool available is the De-Jargonizer, originally designed for scientific communication but might help avoid overspecialized wording.

Best practices

Referencing Gatsby Cloud

Gatsby Cloud is a Gatsby product focused specifically on building and hosting Gatsby sites. There are various parts of the OSS documentation that may benefit from pointing to Gatsby Cloud as a potential platform to explore.

The guidelines for doing so are as follows:

- If possible, Gatsby Cloud should be accompanied by other relevant technologies.
- If Gatsby Cloud does something by default, the docs should still include instructions for accessing that functionality manually.

The spirit of these guidelines is to ensure that users are aware of multiple options for running their Gatsby site. With the exception of `gatsby-cli`, the open source documentation should generally preclude assumptions about technology choices.

Support software versions

When Gatsby commits to support a specific version of software (e.g. Node 8 and up), this is reflected in documentation. Gatsby documentation should be usable by all people on supported software, which means we don't introduce any commands or practices that can't be used by people on versions we've committed to support. In rare circumstances, we'll consider mentioning a newly introduced command or practice as side notes.

Share best practices whenever possible

When there are multiple ways to complete a task, the docs should explain the following:

1. The most fundamental way of completing the task
2. The most common way of completing a task
3. The best way to complete the task on the lowest supported versions of software
4. The best practice and why is it the best (if different than 3)
5. Any tips on how to pick an option

For example, `gatsby-plugin-image` is a component that includes Gatsby best practices for handling images, yet there are more fundamental and common ways of handling them. Documentation ought to make the best practice clear in addition to the most common and fundamental ways.

Docs

"Docs" in Gatsby generally refers to everything under `gatsbyjs.com/docs`.

Docs audience

Developers of all skill levels read the docs and find them useful.

The docs should focus on helping users with the following attributes and goals.

Attributes:

- intermediate to advanced at React
- frontend developer
- prefer using a search engine and/or `ctrl + f` to find things on the gatsbyjs.com site

Looking for:

- way to get a site up and running quickly
- a quick way to get the right words, types, defaults, descriptions, parameters, and returns for the API
- bits of source code to study and/or copy
- step-by-step tutorials for advanced tasks
- understanding how Gatsby works at a deep level, so deep that they could actually modify or customize their own projects, or contribute to Gatsby core
- how does Gatsby really work with Redux, React, and GraphQL?
- error messages that tell them if it's a known bug/issue, lead them to docs, and/or suggest fixes
- guides for how things work in Gatsby
- they usually already have strong opinions or requirements about what they want to use as their CMS or data source and want to know best practices for their workflow
- signs that Gatsby is a reliable, long-term choice (signs that it is growing and improving and evidence it will be around for a long time)
- ways to check their project's requirements against what Gatsby offers
- open source code from well-built example sites

Docs purpose

By referencing the docs, a user should:

- get tasks done as quickly as possible
- evaluate options for getting tasks done as quickly as possible
- build sites and apps as quickly as possible, including the following kinds of sites:
    - marketing
    - blogs
    - portfolio
    - e-commerce
    - authenticated
    - accessible

Docs tone and style

The tone and style of the docs should effectively help the audience reach their goals.

Use personal "you"

The docs use the personal "you" to address the user.

Give experts as much relevant info to get the task done as quickly as possible

Because the audience of the docs is people who have intermediate to expert level of mastery with React, it's important to provide the information needed to complete tasks in addition to all relevant and helpful context, references, and alternatives. The goal: give people the information necessary to get tasks done as quickly and effectively as possible.

In practice, you can reach this goal by two general rules:

- Include an "additional information" section at the bottom of each guide with hyperlinks to relevant external blogposts, tutorials, and other Gatsby resources and docs.

- When there are multiple ways to complete a task, follow these instructions.

Why we chose GitHub for writing and maintaining docs

The way the Gatsby community maintains docs and tutorials must meet the following requirements:

- ability to ship quickly
- ability to iterate quickly
- OSS contributor access
- code editing functionality
- version control
- a way to get feedback on each doc

GitHub meets these requirements.