

# Docs Style Guide

This style guide will help Optimists to write technical content in consistent voice, tone, and style. See the [glossary](#) for an alphabetical listing of commonly-used words, terms, and concepts used throughout the technical docs and across the OP Collective.

- For docs-related questions or comments, create an issue in the [docs repo \(opens in a new tab\)](#)
- .
- For support-related questions or comments, create an issue in the [developers repo \(opens in a new tab\)](#)
- .

## Table of Contents

- [Files, Folders, and Naming Conventions](#)
- [Writing Style](#)
- [Accessibility](#)
- [Content Organization](#)
- [Links](#)
- [Content Types](#)
- [General Formatting](#)

## Files, Folders, and Naming Conventions

### Folder Structure

The folder structure for the [docs.optimism.io \(opens in a new tab\)](#) repository is organized into several high-level categories under the mainpages folder such as builders, chain, connect, and stack.

The left sidebar (leftbar) is managed in the \_meta.json file. You only need to edit the \_meta.json file when you are adding a new page or deleting an existing page. Editing the \_meta.json file too frequently can cause merge conflicts because content is being worked on daily. You will need to accept the changes from other contributors if you are committing a PR.

Don't worry if you're not sure where in the leftbar a new topic belongs. Do your best and when you submit your PR, the Developer Relations team will edit the correct \_meta.json file and determine the right placement.

The right sidebar (rightbar), also known as the page TOC, is created automatically for all the H2 and H3 headings on a page.

### File names

In general, filenames should be as short as possible (~2 to 4 words) and all lower-case. Add a hyphen (-) between each word.

Example: writing-a-guide.mdx or run-node.mdx

### File Paths

File paths should be written the same way as code snippets when used within a docs page, so that they are easier to read. File paths should be wrapped in backticks.

Example: `:/source/docs/assets/images/`

## Writing Style

### Voice and Tone

Write in a friendly, yet professional tone. We are upbeat, knowledgeable, and optimistic about the development of the Optimism Collective, which we try our best to convey in our technical documentation.

### Clear and Concise Language

- Be consistent. Use the same terminology, voice, and tone throughout the documentation.
- Be concise. Focus on providing key information and avoid including superfluous information.
- Use language the audience understands. It is difficult to avoid jargon in the web3 space, but we can do a few things to improve language comprehension for users such as: \* checking the glossary to ensure terminology is clearly defined before using it; and
-

- regularly updating the glossary
- Write in an action-oriented style. Focus on helping the user complete the task at hand by writing in the active voice. An active voice is more direct and reduces ambiguity. Avoid passive voice as it is often vague and creates awkward sentences.
- Avoid gender-specific language. Use the imperative. This form of a verb lets you use the second person (you, your) rather than the third person (him, her, she, his).

## Capitalization

See below for when to use title or sentence case.

- Avoid using all caps as it slows down reading comprehension. If you need to make content prominent, use bold rather than all caps or italics.
- Capitalize proper nouns in sentences.
- Example
  - : I use Visual Studio on my local machine.
- Use title case for all headers (H1, H2, H3) and whenever referring to buttons, tab names, page names, and links within the technical documentation (NOTE: This does not require the words on the button, link, page, or tab to be title case—just when being referring to in the docs). Title case means that you capitalize every word except: articles (a, an, as, the) and coordinating conjunctions (and, or).
- Examples of Title Case:
  - Domains For the Development Environment (header)
  - SelectMake Owner
  - .
  - ClickClear Caches
  - .
  - Select theSettings
  - tab.
- Use sentence case for body content and short phrases, even when the content is a link. Sentence case means you only capitalize the first letter of the sentence.
- Example:
  - If you're trying to figure out how to do something specific as a node operator, you might search our collection [dtutorials](#) or [suggest a new one\(opens in a new tab\)](#)
  - .
- Use lowercase in code snippets by default, unless the code block uses capitalization (e.g., for the name of a function or variable) and you are referring to the function or variable elsewhere within the technical documentation.
- Examples
  - : Rungit add
  - or ImportuseState
  - △
- When in doubt, follow the code base because exact capitalization is necessary in order for the code to compile.

## Accessibility

When creating content, ensure the content you're creating is accessible by a screen-reader, for visually-impaired users, and accessible via mouse or keyboard. For more information regarding accessibility guidelines, see [W3C Web Content Accessibility Guidelines 2.0\(opens in a new tab\)](#) .

### Alt-Text

- Provide alt text for all images so that the screen reader can interpret the purpose of the image and convey that to the user. The alt text should include any content shown in the image so that screen reader can read it to the user.
- Provide alt text for videos by modifying the title attribute of any embedded video content or iframe. The title attribute serves as the alt text for screen readers to provide descriptive information. Video content with speaking or narration must include closed captions.

### Captions

- Provide captions for visual elements/objects whenever possible, so visuals are accessible to all users, regardless of ability. This includes images or screenshots, animated GIFs, promo and tutorial videos, tables, charts, mermaid diagrams, and code blocks.
- Ensure that captions can be translated into major languages.

### Images

- Don't use images of text, code samples, or terminal output. Use actual text.
- Use SVG instead of PNG if available. SVGs stay sharp when users zoom in on the image.

# Content Organization

We aim to use consistent organization that is also user-centered and accessible. This requires intentional work and planning to group technical content into sections instead of using long, dense paragraphs. This also gives readers a visual rest from a usability perspective and improves reading comprehension.

- Use structured headings (H1, H2 or #, ##) to guide readers through the technical documentation.
- Use numbered lists for chronological steps.
- Use bulleted lists to visually separate content that does not require a specific order.
- Format text for optimal readability (bold vs. italics). Avoid using italics in web content as it decreases readability. Use bold if you need to make content prominent but use it sparingly. For instance, **bold**
- is appropriate to use when referring to a specific button or page name in technical documentation.
- Organize technical content to cover only one major concept or task at a time.\* general rule of thumb
- - : documents with > 3 levels of structured headings (H4 or ####) and/or > than 20 minutes estimated reading time (ERT) need revision.
- - revisions will usually require editing for concision, breaking the document apart into multiple pages, or some combination of the two.
- Organize content based on the audience's varying needs and prior knowledge. If pre-requisite knowledge is necessary to complete a task or understand a concept, then share it with users (including links to learn more), so they can easily get up to speed.

## Meta Tags

- Define the metatitle
- ,language
- , anddescription
- for each page to improve SEO ranking of the docs. Place meta tags at the top of the page before the H1 tag, with 3 dashes surrounding the block of text on either side.
- Set the meta title by reusing the H1 page title, but since meta titles are not visually displayed on the page to users, the H1 page title is also required.
- NOTE
- SEO guidelines suggest that meta page titles differ slightly from H1 page titles, but this is handled automatically by our site platform. So, please match H1 page titles to meta page titles for ease of documentation.
- Set the language attribute for accessibility and usability (for screen readers) but also to make it easier to enable language localization support in the future.
- Write meta descriptions as concise overviews (100-150 characters) of the most relevant content of the page.

Example Meta section of docs:

— title: Supercharge Your Dapp with Account Abstraction lang: en-US description: This guide explains how account abstraction enables users to utilize smart contracts to build, onboard, and scale dapps. —

## Page Titles (H1)

- Create concise page titles and format as H1. The title should be able to fit on 1-2 lines.
- Every page must have an H1 heading, in addition to the page title being defined in the SEO meta tags.
- H1 heading is reserved for page titles, so avoid using H1 in any other place on the page.
- For tutorials and quick starts, use task-based page titles starting with gerunds (verb ending in "ing").
- Examples
- :Creating Your Own L2 Rollup
- orRunning a Node

## Headings and TOC

- Use an imperative verb for headings or subheadings in a document, and not a gerund (a verb ending in "ing"). Headings should be shown in H2 tags, and subheadings should be shown in H3 tags.
- Outline the page content according to main topics first. Those will be your headings (tagged as H2). If there are subtopics that belong under a category, display those as subheaders (tagged as H3).
- Example
- :
  - (H1) Supporting OP Mainnet in Your Wallet (H1 page title uses "ing"verb ending)
  - (H2) Connect to OP Mainnet (H2 does not use "ing"verb ending)
  - (H2) Locate Canonical Token Addresses (second H2 does not use "ing"verb ending)
- Use headings in a logical manner, and the site will automatically generate anchor links for H2 and H3 tags and place them in a Table of Contents (TOC) in the right column.
- Avoid H4 levels and above within guide and template pages. As stated elsewhere in this style guide, technical

documents with > 3 levels of structured headings (H4 or ####) usually indicates clarity, organization, or structural issues and should be revised.

- H4 headings are reserved (at this time) for glossary terms. This standardization will make it easier for us to extend glossary functionality across the docs in the future, such as tool tips.

## Listing Prerequisites (Before You Begin)

- Add a "Before You Begin" section at the top of the document if there are tasks a user needs to complete before continuing with the current task, e.g. installing a module, downloading a software update, etc.
- Use the title "Before You Begin" and format as H2. It should follow the page overview/intro section or table of contents.
- Include all the tasks the user needs to complete, including links to aid in usability. Use a bulleted list for more than 2 prerequisite items. Example
  - :
  - (H2) Before You Begin
  - You'll need to enable the ApacheSolr module. Visit the [ApacheSolr \(opens in a new tab\)](#)
  - page on [Drupal.org \(opens in a new tab\)](#)
  - for more information.

## Callouts

- Use callouts to direct users to information necessary to complete a task or information of special importance. When adding a callout to a document, use sentence case and use bold text sparingly.
- Use the correct callout type based on the type of issue: a) info/general, b) warning, c) error. Our documentation platform supports 4 different callout types.\* The default and info callouts are used to share non-sensitive, non-breaking info with users, such as suggestions or best practices that might make the installation or deployment easier.
  - - Example
  - - :
  - - You can easily export your OP Profile data to reuse on other media platforms.
  - - Warning callouts should be used to indicate important info, such as when a product or code will be deprecated.
  - - ⚠
  - - Example
  - - :
  - - This API will be deprecated soon.
  - - Error callouts are reserved for critical issues that cannot be undone or can result in breaking changes, such as when data might be permanently deleted or lost.
  - -
  - - Example
  - - :
  - - This is a dangerous action that permanently deletes your data and cannot be undone. Your data cannot be recovered by Optimism.
- Use callouts sparingly as too many can be confusing to readers. As a general rule of thumb:
- pages with more than 2 callouts likely needs revision and/or a discussion with an Optimism developer or product manager to ensure guide content accuracy.

## Code Samples

- Use code samples as often as possible to help explain concepts. Can be used in guides or tutorials, but every tutorial should have at least one code sample to be useful to developers.
- Any bits of code should be wrapped in backticks and use built-in syntax highlighting. Most documentation platforms automatically apply syntax highlighting when properly defined inside the code block.


Example : This markdown or MDX file

`js console.log('hello, world')` will render this code snippet in javascript with proper syntax highlighting:

`console .log ( 'hello, world' )` \* To improve readability and accessibility even further, consider the following user-centered


options for code blocks:\* adding a filename or title to the code block \* \* adding a caption or description (shown above) \* \* adding or showing the line numbers within the code block (easily refer to a certain code lines within the documentation) \* \* highlighting lines or strings in the code block to draw user's attention to specific areas

## Images, Screenshots, & Icons

- Images, screenshots, and icons are stored in thepublic/img directory in the root folder.
- Every image and screenshot should have descriptive alt text.
- Screenshots should clearly capture the content being discussed in the guide or tutorial.\* Use more than one screenshot if space is an issue and/or to better coordinate screenshots with a particular location or tutorial step in the technical documentation.
- - Use arrows and callouts to help explain the elements in the image that are not already highlighted by the interface. Do not use callouts to highlight the environment and tool, as they are apparent.
- Images can be inserted two ways: embedded in the.mdx file or imported. Use the latter option when you need to add styling to the image, such as a specific height or width, but note that the file path changes when the image is imported.
- File paths to images will vary based on where the image is located and how the image is used (e.g., embedded vs imported into the mdx page — see below for an example).
- Example (embedded)
- : 
- Example (imported)
- :

```
import Image from 'next/image'
```

...

 \* Icons come from[Remix\(opens in a new tab\)](#) \* to maintain consistency across the docs. Use Optimism RedFF0420 \* to color icons before downloading and store icons inpublic/img/icons \* directory.

## Videos

- Use videos sparingly and only for more complex tasks.
- Write meaningful alt text for videos and animations. Include a caption too, if possible.
- Promo videos should be as short as possible, typically no more than 30 seconds.
- Animated gifs should also be short, generally between 10-60 seconds.
- Tutorial videos are considered educational content and should not exceed 10 minutes, based on instructional design best practices.
- Tutorial videos must be hosted by a third party (YouTube, Vimeo, etc.) and include closed captions for accessibility.
- Embed videos with an iframe and add/modify thetitle attribute as needed to make more meaningfulalt-text, which improves accessibility for screen readers.
- Example of video with title for alt-text:

## Links

Developers trust that we will lead them to sites or pages related to their reading content. In order to maintain that trust, it's important that links are transparent, up-to-date, and lead to legitimate resources.

## Internal Links

- Internal links are automatically generated based on the H2 and H3 title tags.
- When linking to a specific H2 or H3 section of a page, the anchor links are always lowercase with dashes taking the place of spaces.
- Example
- : (H3) Test your application
- is converted to test-your-application
- as an anchor link.
- Use anchor links, whenever possible, to guide users to a specific page and location in the technical documentation. This reduces cognitive load and improves overall usability.
- To link to an anchor, such as an H3 tag within a page, you need to append it to the page name preceded by #
- , like this example
- : [any descriptive text](#)
- .

## Linking Across Pages

- Use absolute or relative links when linking across pages in the site. Absolute links are cleaner and easier to work with when pages are nested, so they are the recommended option.
- Examples
- : absolute link (/protocol/deposit-flow)
- versus relative link (../protocol/deposit-flow)
- Use the exact title of the page when linking to it in a sentence, whenever possible, and display it in title case. The link should use default styling with no other formatting (bold, italics, quotations).
- Example
- : Be sure to check out the [OP Stack Getting Started Guide](#)
- to get up to speed.
- Use sentence case when linking to an article without using the exact article title. This causes minimal disruption to the flow of the sentence. The link should use default styling.
- Example
- : For something more advanced, we recommend reading through our page on [sending data between L1 and L2](#)
- .
- Use detailed instructions
- link format to refer users to another article with detailed instructions that are important for completing the current task.
- Example
- : For detailed instructions, see [Article Title](#)
- .
- Use the more information
- link format to guide users to a suggested reading that the user may find helpful because it is related to the task/topic, but not essential for completing the current task.
- Example
- : For more information, see [Article Title](#)
- .

## Content Types

Content types help manage technical content by defining the purpose and common structure for each file type. All content types used in these technical docs have attributes or properties, as defined below.

Document type	Purpose	Examples
Overviews	General introduction to a product or feature, provides a happy-path for readers	<a href="#">Node Operator Overview</a>
Guides	Explain what things are and how they work	<a href="#">Standard Bridge Guide</a>
Quick Start	Briefly explain how to "minimally" get started with a product, often in 30 minutes or less	<a href="#">Superchain App Quick Start</a>
Tutorials	Provide task-oriented guidance with step-by-step "learn by doing" instructions	<a href="#">Bridging ERC-20 tokens with the Optimism SDK</a>
FAQs	Address frequently asked questions	<a href="#">FAQ: OP Mainnet Security Model</a>
Troubleshooting	List common troubleshooting scenarios and solutions	<a href="#">Troubleshooting: Run a Node</a>
Reference	Provide deep, theoretical knowledge of the internal workings of a system, such as API endpoints and specifications	<a href="#">Node and RPC Providers</a>

## Overviews

Overviews provide a general introduction to a product or service and a happy path for readers on how to navigate a particular set of docs or related features; often used at the top-level of a directory, such as `as/builders/dapp-developers/overview` . When done well, overviews accomplish two essential tasks for users:

1. organize
2. the related set of documentation pages to keep developers from getting overloaded by too much information, and
3. establish a 'happy path'
4. to direct developers to the right pages in the documentation set based on their user scenario or use case.

Tables work really well for overview pages, which can both organize page content and establish the happy path for specific developer use cases. Alternatively, headings (H2) can be used, organized by use case, user scenario, or directory page titles.

To maintain consistency, overviews should include all these items:

- overview of what the page will cover
- content organized into discrete sections, by use case, user scenario, or page titles
- clear headings for each section written in parallel style
- next steps section: links to related guides, tutorials, etc.

## Guides

Guides use simple language to explain concepts or complex features, such as what things are and how they work. They are great for on-boarding beginning or novice developers. For developer-focused guides, it is best practice to include illustrations, diagrams, and code samples whenever possible to help explain concepts.

To maintain consistency, guides should include all these items:

- overview of what the guide will cover
- guide content organized into discrete sections
- clear headings for each section written in parallel style
- one or more visual elements (e.g., images, screenshots, illustrations, and/or code samples)
- next steps section: links to related tutorials, other guides, troubleshooting, etc.

## Quick Starts

A Quick Start Guide should be brief (thus "quick"), easy to read, and focused on helping customers get started with only the basics of your product or service. It is usually a condensed version of a longer getting started guide (or a longer tutorial), so it is not a replacement but an accompaniment or companion piece to the docs set. As a rule, this content type should describe only one scenario. It's value lies in its simplicity and ability to speed up onboarding (e.g., installation, deployment, etc.) and developers' first steps, thus improving the developer experience.

To maintain consistency, quick start guides should include these items:

- overview of what the quick start guide will cover and estimated time of how long it will take developers to complete
- guide content organized into discrete sections explaining how to set up the product (e.g., cover minimal setup only)
- clear headings for each section written in parallel style
- next steps section: links to advanced setup options for the product, related tutorials, troubleshooting or error codes, etc.

When writing tutorials or quick starts, steps should be written in parallel style, as sentences with imperative verbs and not gerunds (ending in "ing").

Example :

Step 1:Create Your Site Step 2:Choose Your Framework Step 3:Visit the Dev Environment

## Tutorials

Tutorials are task-oriented pages or videos that include practical, step-by-step instructions for completing a task, activity, or objective. Tutorials are more interactive and hands-on than other technical documentation, often including practical examples, exercises, and demonstrations to help developers learn by doing.

To maintain consistency, tutorials should include all of these items:

- overview of what the tutorial will cover, including context for what problem is being solved
- goals of the tutorial (i.e., what the developer will learn or the finish state)
- before you begin section (also known as prerequisites)
- list of steps involved in the task
- images, screenshots, or code samples (ideally, at least one of these for each step)
- next steps section: links to other tutorials, related guides, troubleshooting, etc.

When writing tutorials or quick starts, steps should be written in parallel style, as sentences with imperative verbs and not gerunds (ending in "ing").

Example :

Step 1:Create Your Site Step 2:Choose Your Framework Step 3:Visit the Dev Environment

## FAQs

FAQs address common questions developers have/might ask about a product, feature, or service. FAQs should be written from the developer's perspective. If there are more than two steps in the answer, use a numbered list. Place FAQs onto their own separate pages, whenever possible, and link to the FAQ from within the respective guide or tutorial.

To maintain consistency, FAQs should include all of these items:

- overview of what product or service the FAQ is about
- questions (usually H2 or H3, depending on page)
- answers (not a heading, usually normal paragraph text beneath the question)

The heading level for FAQs will vary based on if it's an FAQ-only doc or if FAQs are included as part of a larger document. In either case, try not to exceed H3 level when organizing the document.

Example of a simple FAQ

Does Optimism Support ERC-721?

Yes. We have complete and total support for ERC-721 standard. Example of an instructional FAQ

How do I change my Optimism password?

1. SelectSites & Accounts
2. from the user account drop-down menu.
3. Click theAccount
4. tab.
5. SelectChange Password
6. .
7. Enter the information, and clickSave
8. . Include a category heading when you need to group related FAQ content (e.g., See the Optimism Glossary for a detailed example). Category headings are optional, but helpful, for longer FAQs.

## Troubleshooting Guides

Troubleshooting guides list common problems a developer might encounter while using a product or service, identifies the symptoms, and offers solutions to these problems. It is important to accurately capture symptoms produced by the system or interface (e.g., error messages, unexpected page refresh/reload, spinning wheel, etc.), so developers know if their system response aligns with one of the common problems identified in the troubleshooting guide.

A troubleshooting guide should be limited to identifying common problems for one particular product or service.

To maintain consistency, troubleshooting guides should include all of these items:

- overview of what product or service the troubleshooting guide will cover
- common problem (usually H2 or H3, depending on page)\* cause of problem
- - symptom of problem (system response, captured as image/screenshot)
- solution (step-by-step)
- next steps section: links to related tutorials, other guides, etc.

## Technical Reference

Technical references provide deep, theoretical knowledge of the internal workings of a system. These often come in the form of requirements or system specifications developers need to run the product efficiently, so lists and tables, such as API endpoints and error codes are commonplace. A technical reference page is usually quite long, so it is best practice to embed a table of contents (TOC) at the top of the page to help organize material for developers. From a usability perspective, this practice shows developers what will be covered in the reference in advance, and allows them to jump to a specific section, if desired.

To maintain consistency, technical references should include all these items:

- overview of what the reference will cover
- table of contents
- reference content organized into discrete sections, with parallel headings
- one or more visual elements (e.g., flow diagrams, illustrations, and/or code samples)
- suggestions for further reading (links spread throughout the reference doc)

Technical references often include more links throughout the document than other content types, often linking to other technical references, guides, tutorials, glossary definitions, etc. Since the purpose of technical reference material is to



educate developers on a deeper level about the topic of their choosing, this is a common and expected practice and is a good indication of a strong technical reference.

## General Formatting

### Fonts

Fonts in Optimism technical documentation are setup to follow brand guidelines established by marketing (e.g., heading fonts are different than body or paragraph font). Please do not change them.

### Bullets & Unordered Lists

Please use\* instead of- for items in a list. This maintains consistency across the docs.

### Date & Numbers

- Use the full month, day, year format for dates whenever possible. Do not abbreviate the month. In a form or when space is limited, use slashes in the format of month/day/year without any leading zeros.
- Examples
  - :January 10, 2014
  - or1/10/2014
- Spell out all numbers under 10. For numbers 10 and above, use the numeral.
- Example
  - : CompanyX operates five nodes and plans to add 12 more.

### Abbreviations

- Use contractions with intention. Contractions can be used to create a conversational, informal tone, such as in FAQs or Tutorials. Avoid using contractions in UI labels (i.e. button names, page headers, etc), error messages/error codes, or interactive page elements.
- Avoid abbreviating common words. It is preferable to spell it out unless there are major space limitations, such as in a table.
- Example
  - :Use account
  - (not acct.) andnumber
  - (not no.)
- Spell out acronyms the first time used on any given page. Then, abbreviate in parentheses afterward. Link users to the glossary, when applicable.
- Example
  - :Externally Owned Account (EOA)

### Punctuation

- Ampersand (&)
- Only use "&" in headings where items are grouped or in proper names. Do not use in sentences.
- Colon (:)
- Use to introduce a list or series.
- Commas (,)
- Use a serial comma in lists of three or more items and use the oxford comma preceding the "and" before the last element in a list.
- Example
  - : The developer built a node, social app, and DeFi app for the Optimism Collective.
- Em dash (—)
- Use to indicate a break in thought or a parenthetical comment. Do not add spaces around the em dash.
- Example
  - : The developer graduated—with honors—from Optimism Bootcamp.
- En dash (–)
- Use to indicate a range or a continuation of a series. Use spaces on each side of the en dash.
- Example
  - :Pages 11 – 19
  - orMon – Fri
  - orNov 1 – 17
- Exclamation point (!)
- Avoid. We do not use exclamation points in user assistance content. It is more appropriate for marketing and sales content, but not in the UI or technical documentation.
- Hyphen (-)
- Use to connect two words. No spaces are needed around the hyphen.

- Example
- :developer-focused product
- orcompany-wide holiday
- Slash (/)
- Avoid using as the slash is reserved for file names (see above). Use "or," "and," or "or both" instead.

[Contribute to OP Stack Changelog](#)