



Strapi Documentation Style Guide



The Strapi documentation is based on Docusaurus, and the formatting options are a mix of basic Markdown or HTML, and custom React components. Below are listed all these content formatting options as well as guidelines and best practices to follow when writing for Strapi's [User Guide](#), [Developer Documentation](#), or [Strapi Cloud Documentation](#).

Content formatting

[Titles](#)

[Text highlights](#)

[Callouts/Admonitions](#)

[Links](#)

[Lists](#)

[Code examples](#)

[Details](#)

[Tables](#)

[Tabs](#)

[Requests & Responses](#)

Specific formatting

[Badges](#)

[Snippets](#)

Illustrations

[Screenshots](#)

[Emojis](#)

[Icons](#)

Content formatting

Titles

MARKDOWN

Titles are the base of a documentation structure, as they shape the content by separating it into coherent sections.

There is only one H1 title, at the very beginning of the documentation file. Other titles can be used several times throughout the file but must keep a logical order (e.g. H3 shouldn't be used if there is no H2 before).

```
# H1 title (displayed by default in main navigation)

## H2 title (displayed by default in main navigation)

### H3 title (displayed by default in secondary navigation)

#### H4 title (displayed by default in secondary navigation)

##### H5 title (by default not displayed in any navigation — should be avoided)
```



Best practices for titles

- Titles should be short, yet give a clear idea of what topic will be tackled in the related documentation section.
- Titles shouldn't be a question.
- Titles should all have a similar format, especially titles of the same level.
- Titles can emphasise the action that is tackled in the following documentation by starting with an action verb (e.g. "Writing content", "Creating content-types").

Text highlights

MARKDOWN

Text highlights are basic formatting that represent and/or highlight items from user interfaces.

- *italics*: for admin panel section names, window and field names
- **bold**: for button names
- `code style` : for small code excerpts, file names and paths (Developer Documentation)

```
*italics*

**bold**

`code style`
```

Callouts/Admonitions

REACT COMPONENT


Callouts (also called admonitions) highlight information by separating it from the regular, main content.

From lowest to highest level of importance:

- **Tips**: for tips and tricks, useful information but not required to understand how Strapi works



TIP

Click the search icon  in the sub navigation to use a text search and find one of your content-types more quickly!

- **Notes**: for helpful, time-saving information



NOTE

New entries are only considered created once some of their content has been written and saved once. Only then will the new entry be listed in the list view.

- **Prerequisites**: for necessary conditions allowing to successfully complete a full procedure



There must be **no space** between `:::` and the name of the callout.

```
:::tip
[content goes here]
:::
```

```
:::note
[content goes here]
:::
```

```
:::prerequisites
[content goes here]
:::
```

```
:::caution
[content goes here]
:::
```

✓ **Prerequisites**
Before installing Strapi, the following requirements must be installed on your computer:

- **Node.js:** Only Maintenance and LTS versions are supported (`v14` , `v16` , and `v18`).
 - Node v18.x is recommended for Strapi `v4.3.9` and above
 - Node v16.x is recommended for Strapi `v4.0.x` to `v4.3.8`.
- Your preferred Node.js package manager:
 - `npm` (`v6` only)
 - `yarn`
- **Python** (if using a SQLite database)

- **Cautions:** for important information such as mistakes prevention, recommendations, unstable or unreliable documentation

⚠ **CAUTION**
In order to access the admin panel, your Strapi application must be launched, and you must be aware of the URL to its admin panel (e.g. `api.example.com/admin`).

- **Warnings:** for highly important information such as data loss, project crash prevention, unsupported documentation

🔥 **WARNING**
It is a security risk to expose static session middleware keys in a deployed environment. An `.env` file or environment variables should be used instead.

▶ Example: sessions keys in .env file

- **Additional callout:** the Strapi special

For links to other Strapi resources, aiming at promoting the Strapi ecosystem in a more marketing-like way. It should only be used by the Strapi team.

👋 **Welcome to the Strapi community!**
If you have any trouble with your Strapi experience, you can reach us through [GitHub](#) or our [forum](#)! The Strapi Community and Strapi team are always available to answer your questions or help you with anything!

```

:::warning
[content goes here]
:::

```

```

:::strapi
[content goes here]
:::

```

Links

MARKDOWN

Links can redirect users to other parts/sections of <https://docs.strapi.io/> (internal links) and to any other resource outside the Strapi documentation (external links).

Each collection type is divided into 2 interfaces: the list view and the edit view (see [\[Writing content\]\(writing-content\)](#)).

Sorting can be enabled for most fields displayed in the list view table (see [\[Configuring the views of a content-type\]\(/user-docs/content-manager/configuring-view-of-content-type\)](#)).

All formatting options are in the [\[style guide\]\(https://www.notion.so/strapi/Strapi-docs-style-guide-b9c21387732741ed8b0a3f9701612577\)](https://www.notion.so/strapi/Strapi-docs-style-guide-b9c21387732741ed8b0a3f9701612577).



Best practices for links

- If the name of the doc/section to link is already in the sentence, directly put the link on that doc/section name. Otherwise, add the name of the related doc + the link in parenthesis in the following format: `(see [doc name](link))`.
- Consider links as IDs not paths: do not add the extension at the end of the file name (e.g. `writing-content.md` → `writing-content`).
- If a file had a `slug` property defined, use its value as ID.
- Use absolute links instead of relative ones for more stability & reliability (e.g. `../content-manager/configuring-view-of-content-type` → `/user-docs/content-manager/configuring-view-of-content-type`).

Lists

MARKDOWN

Lists display information in a clearer, more readable way than in an enumeration or full paragraph.

- Bullet lists are for same-level items. They are used to avoid enumerations.
- Numbered lists are for procedures. Each number describes an action that will lead the user to complete a goal.

If the list contains too many items (more than 10) and/or items need additional explanations, use a [table](#) instead.

This is the beginning of my sentence which:

- has a first point,
- and a second point.

It is also possible to write my sentence differently.

- A possible way is as shown below.
- Another possible way is like so.

To write a piece of documentation properly, you should:

1. Open Notion.
2. Go to the Documentation Handbook.
3. Read the Strapi docs style guide.



Best practices for lists

- For bullet lists — If the bullet points are entirely part of a same sentence, avoid starting with a capital letter and end the point with either a comma or nothing. If the bullet points are independent small sentences, the basic grammar rules apply (capital letter and full stop).
- For numbered lists — Always start with a capital letter and end with a full stop.

Code examples

REACT COMPONENT

Code examples and command lines are displayed in a terminal-like style, that also contains a "Copy" button for easier use.

It is possible to add a title to the code example, to indicate the file where the code is/should be located/written.

```
```language
[code goes here]
```
```

```
```language title="./src/api/[api-name]/content-types/resta
[code goes here]
```
```

There are 2 options for code examples:

- Simple:

```
strapi develop
options: [ --no-build | --watch-admin | --browser ]
```

- Advanced, which are mixed with `tabs` to display various language/environment options:

Yarn NPM

```
yarn create strapi-app my-project
# 'yarn create' creates a new project
# 'strapi-app' is the Strapi package
# 'my-project' is the name of your Strapi project
```

```
<Tabs groupId="GROUP-NAME">

<TabItem value="NAME-TAB-1" label="LABEL-1">
  ```language
 [code goes here]
  ```
</TabItem>

<TabItem value="NAME-TAB-2" label="LABEL-2">
  ```language
 [code goes here]
  ```
</TabItem>

</Tabs>
```



Best practices for code examples

For better cross-page syncing, use the following `groupId` and `value` properties for advanced code examples:

- Use case: YARN vs. NPM
 - `groupId` : `yarn-npm`
 - `value` properties : `yarn` and `npm`
- Use case: JavaScript vs. TypeScript
 - `groupId` : `js-ts`
 - `value` properties : `js` and `ts`
- Use case: axios vs. fetch
 - `groupId` : `axios-fetch`
 - `value` properties : `axios` and `fetch`

Details

HTML

Details are blocks that by default only display a title, as the rest of the content is collapsed. It is used for additional examples that are not essential and can therefore be hidden by default to make the documentation shorter and cleaner.

► JavaScript query (built with the qs library):

```
<details>
<summary>Title of the details block</summary>

  ``` [code goes here]
  ```

</details>
```

▼ JavaScript query (built with the qs library):

`qs` can be used to build the query URL used in the example above:

```
const qs = require('qs');
const query = qs.stringify(
  {
    fields: ['title', 'body'],
  },
  {
    encodeValuesOnly: true, // prettify URL
  }
);

await request(`/api/users?${query}`);
```

Tables

MARKDOWN

Tables display more clearly complex groups of items related to the same kind of information. They can also replace lists if they become too long (more than 10 items).

In the User Guide, tables are used to document all fields and options of the same interface.

Setting name	Instructions
Label	Write the label to be used for the field in the list view table.
Enable sort on this field	Click on ON or OFF to able or disable the sort on the field.

Title 1	Title 2	...
-----	-----	-----
Item 1	Item 1	...
Item 2	Item 2	...
Item 3	Item 3	...

Tabs

REACT COMPONENT

Tabs tables display concisely one same kind of information available for several use cases.

[GitHub](#) [Facebook](#) [Google](#) [AWS](#) [Twitter](#) [Discord](#) [Twitch](#) [Instagram](#) [VK](#)

Cognito

Using ngrok

Github doesn't accept `localhost` urls.
Use `ngrok` to serve the backend app.

ngrok http 1337

Don't forget to update the server url in the backend config file `config/server.js` and the server url in your frontend app (environment variable `REACT_APP_BACKEND_URL` if you use `react login example app`) with the generated ngrok url.

```
<Tabs groupId="GROUP-NAME">

<TabItem value="NAME-TAB-1" label="LABEL-1">
[content goes here]
</TabItem>

<TabItem value="NAME-TAB-2" label="LABEL-2">
[content goes here]
</TabItem>

</Tabs>
```

Requests & Responses

REACT COMPONENT

This component displays API requests and responses examples in an API documentation-like style.

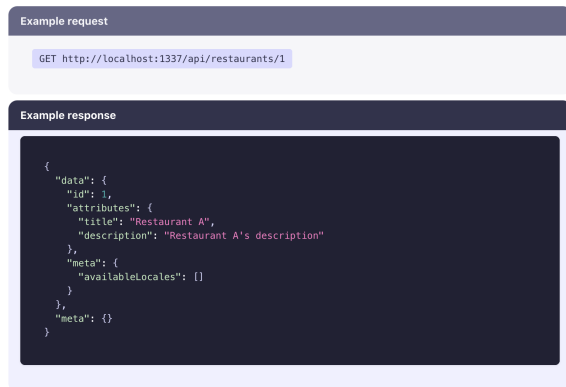
`Request` and `Response` can be used in 2 ways:

```
<ApiCall>

<Request>
```

- as standalone components,
- or wrapped in an `ApiCall` component.

Only the `ApiCall` component is responsive (i.e. the layout is side-by-side only on large viewports while remaining vertical on smaller viewports).



```

`request goes here`

</Request>

<Response>

  `` language
  [code goes here]
  ...

</Response>

</ApiCall>

```

Specific formatting

Badges

REACT COMPONENT

Badges are added to a title to indicate the specificity of the following content/described feature.

There are 2 badges options:

- Pricing badges, to indicate the features and actions restricted to the Enterprise plan, as opposed to the free Community plan.
- Release badges, to notify users if a feature is in alpha or beta.

```
<EnterpriseBadge />
```

```

<AlphaBadge />
<BetaBadge />

```

Managing Review Workflows Enterprise Alpha

The Review Workflows feature allows you to create and manage any desired review stages for your content, enabling your team to collaborate in the content creation flow from draft to publication. Configured review workflows are available in the [Content Manager](#) and [Content-Types Builder](#).

Snippets

REACT COMPONENT

Snippets allow effectively reusing identical content in several places and/or files. They make updates easier and more reliable, since they are done from one single file instead of several.

Snippets are stored in the `docusaurus/docs/snippets` folder.

```

import SnippetName from '../docs/snippets/snippet-file-name.m

(...)

<SnippetName components={props.components} />

```



Best practices for snippets

- Import the snippet at the beginning of the documentation file, right before the H1 title.
- The name of the snippet in the page can be decided by the writer, as long as it remains the same throughout the whole file.

Illustrations

Screenshots

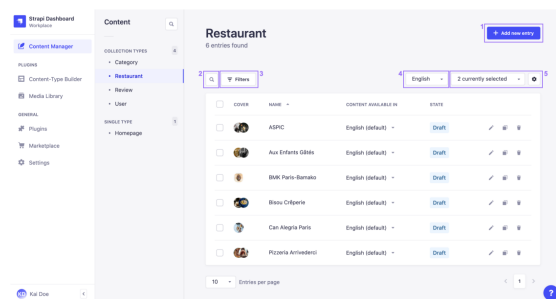
Screenshots are mostly used in the User Guide. They are generally displayed after an introduction-type sentence, and before the explanatory content. One screenshot per section is generally enough.

All screenshots are stored in the `static/img/assets` folder at the root of the Docusaurus documentation folder. They are grouped in various sub-folders named after documentation sections or features.



Nimbus Screenshot is a nice extension to make screenshots and edit/annotate them.

```
! [image description] (/img/assets/subfolder/file_name.png)
```



From the list view, it is possible to:

- create a new entry (1),
- make a textual search (2) or set filters (3) to find specific entries,
- if the [Internationalization plugin](#) is installed, filter by locale to display only the entries translated in a chosen locale (4),
- configure the fields displayed in the table of the list view (5),
- edit (see [Writing content](#)), duplicate (see [Deleting content](#)) or delete (see [Deleting content](#)) an entry.



Best practices for screenshots

- Prefer screenshots of the whole interface instead of cutting only part of it. It is easier for the user to locate where something is on the interface using the context.
- To indicate a specific area of the screenshot that's mentioned in the documentation: edit the screenshot to add a rectangular shape around the area. The shape should be Strapi purple (#8c4bff), and its lines should be 3px wide.

Emojis

Emojis can be used but very sparingly. Avoid using emojis in all documentations. Instead, prefer using them:

- as icons in special notices (default),
- as icons in section titles,
- in more friendly types of documentation (docs sections introductions, docs about Strapi as a company, quick start guides, guideline docs targeting the Community, etc.)

CONGRATULATIONS!

You have just created a new Strapi project! You can start playing with Strapi and discover the product by yourself using our [User Guide](#), or proceed to part B below.

Part B: Build your content



The installation script has just created an empty project. We will now guide you through creating a restaurants directory, inspired by our [FoodAdvisor](#) example application.

Icons

Icons are used to replicate those used on Strapi user interfaces (e.g. the admin panel), to help users locate them on the interface. Icons are available in the `static/img/assets` folder, in the SVG format, and should be integrated in the documentation files as images.



For other usages than mimicking Strapi user interfaces icons, it is possible to use FontAwesome icons: `<Fa-IconName />`

- Below the previous editing options, a table (3) lists all the fields created and configured for the content-type or component. From the fields table, it is possible to:
 - Click on the edit button  to edit a field
 - Click on the delete button  to delete a field

```
![IconName icon](/img/assets/icons/icon.svg)
```