monodocs Sample

Contents

Overview 4
- Build / preview 4
- Features you can check on this site 4

markdown 5
- GFM samples 5
  - Headings (ATX) 5
  - Level 2 6
  - Headings (Setext) 6
  - Small heading (Setext H2) 6
  - Paragraphs and line breaks 6
  - Horizontal rule 7
  - Blockquotes 7
  - Escapes and entity references 8
- Text formatting 9
  - Emphasis 9
  - Links 9
  - Autolinks (GFM extension) 10
  - Images 10
- Lists 12
  - Bullet lists (unordered) 12
  - Numbered lists (ordered) 12
  - Task lists (GFM extension) 13
  - Loose and tight lists 14
  - Multiple blocks in one item 14
- Code and tables 16
  - Fenced code blocks 16
  - No language / indented code 17
  - Tables (GFM extension) 17
  - Mermaid 18
- Footnotes 19
  - Footnotes 19
- Alerts (GFM alerts) 20
  - The five alert types 20
  - Multiple paragraphs and other syntax 21
- Wide content 22
  - A wide table 22
  - A code block with long lines 22
  - A large image 23

Mermaid (a wide diagram) 23
Math 24
Inline 24
Display 24
Matrices and aligned equations 25
advanced 26
Advanced (nested hierarchy) 26
Links follow even from deep folders 26
Deep headings and TOC 27
Level 2: Overview 27
Level 2: Summary 27
asciidoc 28
AsciiDoc samples 28
Section headings 28
Paragraphs and line breaks 28
Text formatting 29
Replacements (text substitution) 30
Comments 30
Lists 31
Bullet lists (unordered) 31
Numbered lists (ordered) 31
Checklists 32
Description lists 32
Terms and nesting 33
Multiple blocks in a list item 34
Blocks 35
Source blocks (syntax highlighting) 35
Listing blocks / literal blocks 36
Example blocks 36
Sidebar blocks 37
Quote blocks / verse blocks 37
Admonitions 38
Mermaid ([source,mermaid]) 40
Passthrough (raw HTML) 40
Tables 42
A basic table 42
Cell formatting and multiple lines 43
CSV format 43
Links and images 45
Links 45
Attributes 45
Internal anchors and cross-references 46

Images 46
UI macros 47
Footnotes 47
Wide content 48
A wide table 48
A listing with long lines 48
A large image 49
mixed 50
Mixed (same folder) 50
Anchor target heading 50
Companion (AsciiDoc) 51
Anchor example 51
PDF output 52
What the PDF includes 52
Configuration (monodocs.config.yml) 53
Runtime requirements 53
Search 54
Matching 54
Narrowing with several keywords 54
Results that matched a heading 54
Keyboard 55
Highlighting in the body 55
Loose matching 55

Overview

🌐 **English** · 日本語

A sample that gathers monodocs' supported syntax and features into **a single site**. Open each category from the folders in the left sidebar.

Every syntax page shows the **source (raw Markdown / AsciiDoc)** first, then the **rendered output (HTML)** below it, so the correspondence between markup and rendering is clear at a glance.

- **Markdown (GFM)**: Markdown samples
- **AsciiDoc**: AsciiDoc samples
- **Mixed (same folder)**: Markdown / AsciiDoc mixed

The 2nd is a link from Markdown to an AsciiDoc page. Cross-format links are also converted into hash routes within the single HTML (mixed support).

Build / preview

```
monodocs build examples/en -o dist/sample.html   # single HTML
monodocs build examples/en --format pdf -o dist/sample.pdf   # PDF
monodocs serve examples/en            # http://127.0.0.1:4173/
```

For **PDF output** via the single HTML (bookmarks, in-content links, font requirements, and more), see PDF output.

Search collects queries you can type against this sample to see how matching, keyboard navigation, and body highlighting behave.

Features you can check on this site

- Full-text search / in-page table of contents (scroll-synced) / prev–next navigation
- Dark mode / sidebar collapse & auto-expand
- Code highlighting (shiki) / Mermaid diagrams / Math (MathML)
- Image data URI embedding / cross-file links & AsciiDoc xref / footnotes
- Tables, code, images, and diagrams wider than the body width (Markdown / AsciiDoc)
- Full vertical expansion of all pages when printing / PDF output (with bookmarks and clickable links)

GitHub Flavored Markdown samples

This sample covers the main syntax of GitHub Flavored Markdown (CommonMark + GFM extensions) to check its rendering in `monodocs`.

Each item shows the **source (raw Markdown)** first, then the **rendered output (HTML)** below it.

Build example:

```
monodocs build examples/en -o dist/sample.html
```

Open each category from the left sidebar: Text formatting / Lists / Code and tables / Footnotes and link references / Alerts / Wide content / Math / Advanced (nested hierarchy).

Headings (ATX)

One to six `#` characters set the level.

Source:

```
## Level 2

### Level 3

#### Level 4

##### Level 5

###### Level 6
```

Rendered:

Level 2

Level 3

Level 4

Level 5

Level 6

Headings (Setext)

`===` / `---` underlines also create headings (H1 / H2 only).

Source:

```
Big heading (Setext H1)
=======================

Small heading (Setext H2)
-------------------------
```

Rendered:

Big heading (Setext H1)

Small heading (Setext H2)

Paragraphs and line breaks

A blank line separates paragraphs. A hard line break is a trailing backslash `\` (or two invisible trailing spaces). Here we show the visible, easy-to-read backslash example.

Source:

```
This is a paragraph. Blank lines separate paragraphs.

A backslash forces a line break.\
And here we are on the next line.
```

Rendered:

This is a paragraph. Blank lines separate paragraphs.

A backslash forces a line break.
And here we are on the next line.

Horizontal rule

`---` / `***` / `___` all create a horizontal rule.

Source:

```
A section divider.

---
```

Rendered:

A section divider.

Blockquotes

`>` creates a blockquote; nesting and lists inside quotes work too.

Source:

```
> A blockquote.
>
> > A nested blockquote.
>
> - A list inside a quote
> - Second item
```

Rendered:

> A blockquote.
>
> > A nested blockquote.
> >
> > - A list inside a quote
> > - Second item

Escapes and entity references

Backslashes escape symbols, and HTML entity references work too.

Source:

```
Escape symbols with a backslash: \*not italic\* / \`not code\` / \# not
heading

Entity references: &copy; &amp; &lt; &gt; &#169; &hearts;
```

Rendered:

Escape symbols with a backslash: *not italic* / `not code` / # not heading

Entity references: © & < > © ♥

Text formatting, links, and images

Each item shows the **source** first, then the **rendered output (HTML)** below it.

Emphasis

Decorations such as italic, bold, strikethrough, and inline code.

Source:

```
- Italic: *asterisks* / _underscores_
- Bold: **double asterisks** / __double underscores__
- Bold italic: ***triple*** / **_mixed_**
- Strikethrough (GFM): ~~struck out~~
- Inline code: `const x = 1;` / when it contains a backtick: `` `code` ``
```

Rendered:

- Italic: *asterisks* / *underscores*
- Bold: **double asterisks** / **double underscores**
- Bold italic: ***triple*** / ***mixed***
- Strikethrough (GFM): ~~struck out~~
- Inline code: `const x = 1;` / when it contains a backtick: `` `code` ``

Links

Inline, with title, reference-style, and same-site links.

Source:

```
- Inline: [monodocs](https://example.com/monodocs)
- With title: [hover to see](https://example.com "This is a title")
- Reference link: [reference style][ref] and [collapsed][]
- Another page in the same site: [the lists page](lists.md)

[ref]: https://example.com/reference
[collapsed]: https://example.com/collapsed
```

Rendered:

- Inline: monodocs
- With title: hover to see
- Reference link: reference style and collapsed
- Another page in the same site: the lists page

Autolinks (GFM extension)

URLs and email addresses become links automatically.

Source:

```
- A bare URL: https://github.github.com/gfm/
- In angle brackets: <https://example.com>
- Starting with `www.`: www.example.com
- Email: <docs@example.com> / contact@example.com
```

Rendered:

- A bare URL: https://github.github.com/gfm/
- In angle brackets: https://example.com
- Starting with `www.`: www.example.com
- Email: docs@example.com / contact@example.com

Images

Inline images and reference-style images (images are embedded into the single HTML as data URIs).

Source:

```
![monodocs logo](images/logo.svg "Logo")

A reference-style image:

![reference image][logo]

[logo]: images/logo.svg "Reference logo"
```

Rendered:



A reference-style image:



Lists

Each item shows the **source** first, then the **rendered output (HTML)** below it.

Bullet lists (unordered)

`-` / `*` / `+` can all be used as markers. Indentation creates nesting.

Source:

```
- Item A
- Item B
  - Nested B-1
  - Nested B-2
    - Even deeper
- Item C
```

Rendered:

- Item A
- Item B
  - Nested B-1
  - Nested B-2
    - Even deeper
- Item C

Numbered lists (ordered)

You can also set the starting number. Two number lists separated by a paragraph become distinct lists, and the second starts from the number you specify (a blank line alone continues the same list).

Source:

```
1. First
2. Next
3. After that
   1. Nested 3-1
   2. Nested 3-2

Specify a start number (separated by a paragraph to make a new list):

5. From 5
6. 6
7. 7
```

Rendered:

1. First
2. Next
3. After that
    1. Nested 3-1
    2. Nested 3-2

Specify a start number (separated by a paragraph to make a new list):

5. From 5
6. 6
7. 7

Task lists (GFM extension)

`- [ ]` / `- [x]` create checkboxes.

Source:

```
- [x] A completed task
- [ ] An incomplete task
  - [x] A child task (done)
  - [ ] A child task (not done)
```

Rendered:

- [x] A completed task
- [ ] An incomplete task
    - [x] A child task (done)

    - [ ] A child task (not done)

Loose and tight lists

When there are blank lines between items, the list becomes "loose" and each item is rendered as a paragraph.

Source:

```
A tight list:

- Lines are close together
- Second line

A loose list (blank lines between items):

- Rendered as paragraphs

- A blank line between items makes it loose
```

Rendered:

A tight list:

- Lines are close together
- Second line

A loose list (blank lines between items):

- Rendered as paragraphs

- A blank line between items makes it loose

Multiple blocks in one item

You can continue paragraphs or quotes within a list item by matching the indentation.

Source:

```
- An item that contains multiple paragraphs.

  This is the second paragraph of the same item (indented).

  > A quote inside the item.
```

Rendered:

- An item that contains multiple paragraphs.

  This is the second paragraph of the same item (indented).

  > A quote inside the item.

Code blocks and tables

Each item shows the **source** first, then the **rendered output (HTML)** below it.

Fenced code blocks

When you specify a language, it is syntax-highlighted by shiki.

Source:

````
```js
// JavaScript
function greet(name) {
  return `Hello, ${name}!`;
}
console.log(greet("monodocs"));
```
````

Rendered:

```js
// JavaScript
function greet(name) {
  return `Hello, ${name}!`;
}
console.log(greet("monodocs"));
```

Tilde (`~~~`) fences work too.

Source:

````
~~~yaml
title: tilde fence
nested: true
~~~
````

Rendered:

```
title: tilde fence
nested: true
```

No language / indented code

A fence with no language, or four leading spaces, also creates a code block (without highlighting).

Source:

````
```
A code block with no language specified (no highlighting)
```

    indented code block
    second line
````

Rendered:

```
A code block with no language specified (no highlighting)
```

```
indented code block
second line
```

Tables (GFM extension)

`:---` left-aligns, `:---:` centers, `---:` right-aligns. You can use `**emphasis**` and `` `code` `` inside cells.

Source:

```
| Feature           | Markdown | AsciiDoc |
| ----------------- | :------: | -------: |
| Headings          |    ✅    |       ✅ |
| Tables            |    ✅    |       ✅ |
| Footnotes         |    ✅    |       ✅ |
| Code highlighting |  shiki   |    shiki |
```

Rendered:

Feature	Markdown	AsciiDoc
Headings	✅	✅
Tables	✅	✅
Footnotes	✅	✅
Code highlighting	shiki	shiki

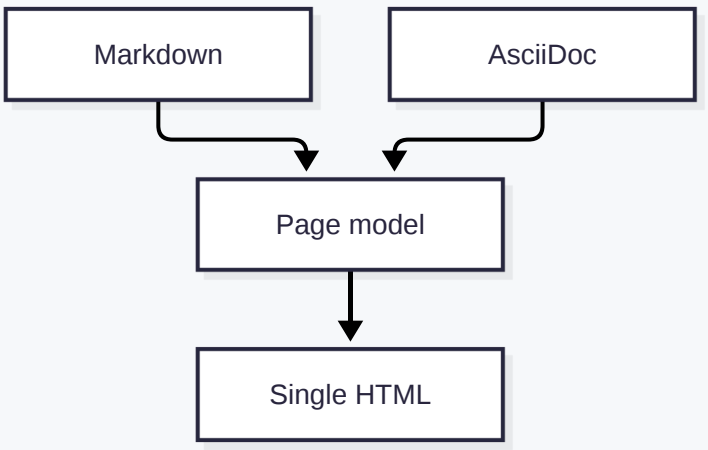
Mermaid

A ```` ```mermaid ```` fence is rendered as a diagram.

Source:

````
```mermaid
graph TD
  A[Markdown] --> B[Page model]
  C[AsciiDoc] --> B
  B --> D[Single HTML]
```
````

Rendered:



Footnotes (GFM extension)

`[^id]` creates a footnote reference and `[^id]: ...` writes its body. Footnote IDs are made unique per page within the single HTML, so they never collide across pages. Clicking a reference or a back-link scrolls to the spot without switching pages.

Source:

```
You can place footnote references in the text[^1]. You can place
several[^note].
You can also reuse the same footnote[^1]. Long notes are fine too[^long].

[^1]: This is the first footnote.
[^note]: A named-label footnote. It can include **emphasis**, `code`, and a
[link](https://example.com).
[^long]:
    A multi-paragraph footnote.

    Continue the indentation to add a second paragraph.
```

Rendered:

You can place footnote references in the text[1]. You can place several[2]. You can also reuse the same footnote[1]. Long notes are fine too[3].

Footnotes

1. This is the first footnote. ↩ ↩[2]
2. A named-label footnote. It can include **emphasis**, `code`, and a link. ↩
3. A multi-paragraph footnote.

   Continue the indentation to add a second paragraph. ↩

Alerts (GitHub alerts)

Placing a marker such as `[!NOTE]` at the start of a blockquote turns it into an alert. There are five types — **NOTE / TIP / IMPORTANT / WARNING / CAUTION** — rendered with the same structure and colors as AsciiDoc admonitions (compare with AsciiDoc Admonitions).

Each item shows the **source** first, then the **rendered output (HTML)** below it.

The five alert types

Put the marker alone on the first line of the blockquote (if you continue text on the same line, it stays an ordinary quote).

Source:

```
> [!NOTE]
> Supplementary information useful for the reader to know.

> [!TIP]
> A small piece of advice or a shortcut.

> [!IMPORTANT]
> Key information essential to achieving the goal.

> [!WARNING]
> Something that needs attention; overlooking it may cause problems.

> [!CAUTION]
> A warning about operations with undesirable results or risks.
```

Rendered:

> **Note**
> Supplementary information useful for the reader to know.

> **Tip**
> A small piece of advice or a shortcut.

> **Important**
>
> Key information essential to achieving the goal.

> **Warning**
>
> Something that needs attention; overlooking it may cause problems.

> **Caution**
>
> A warning about operations with undesirable results or risks.

Multiple paragraphs and other syntax

After the marker line, you can continue with paragraphs, lists, code, and so on.

Source:

```
> [!NOTE]
> The first paragraph.
>
> The second paragraph can include **emphasis**, `code`, and lists.
>
> - Bullet 1
> - Bullet 2
```

Rendered:

> **Note**
>
> The first paragraph.
>
> The second paragraph can include **emphasis**, `code`, and lists.
>
> - Bullet 1
> - Bullet 2

Wide content

The body is kept at a comfortable reading width (max 860px by default). You can change the fixed width with `html.contentWidth` in the config file, or set it to `full` to span the full width. Set `html.contentWidthToggle` to `false` when readers should not be able to change that width. Set `html.contentWidthDefault` to `wide` to start in wide mode until the reader makes a choice. This sample shows how tables, code blocks, images, and diagrams wider than that are displayed.

> **Note**
> In the default theme, the body width can be adjusted with `html.contentWidth`. Tables and code that don't fit the body width can be **scrolled horizontally within that element** to view the whole thing. Images and diagrams are **scaled down to fit the body width** (no horizontal scrollbar appears for the whole page).

A wide table

A table with many columns won't fit the body width. The whole table becomes a scroll area, and you can **scroll horizontally over the table** to see the remaining columns.

Metric	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
Page views	1200	1840	2010	1750	2230	2680	3120	2990	2640	2880	3310	3720
Unique	820	1190	1320	1180	1460	1710	1980	1920	1700	1840	2100	2350
Bounce (%)	58	55	53	54	51	49	47	48	50	49	46	44

A code block with long lines

Code containing long lines **scrolls horizontally within the code block**. Short code is shown as-is.

Long line:

```
docker run --rm -it --name monodocs-dev -v "$(pwd)":/work -w /work/app -e
NODE_ENV=development -p 4173:4173 monodocs-dev node
packages/cli/dist/index.js serve ../examples/en --host 0.0.0.0 --port 4173
```

Short line:

```
const width = 860;
```

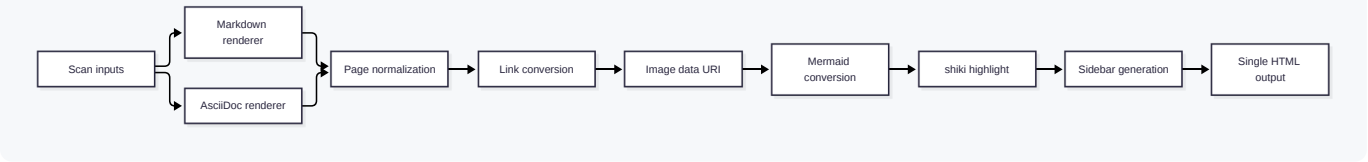
A large image

An image whose intrinsic width is larger than the body width is **scaled down to fit the body width** (`max-width: 100%`).



Mermaid (a wide diagram)

A horizontally long flow diagram is also displayed to fit the body width.



Math

`$...$` writes a formula inside a sentence, and `$$...$$` or a fenced `math` block writes one on its own line. Formulas are rendered to MathML at build time, so the output needs no script.

Each item shows the **source** first, then the **rendered output (HTML)** below it.

> **Note**
> The browser draws formulas with an OpenType MATH font. In the PDF, monodocs sets them in a font it carries, New Computer Modern Math, on any machine. In the HTML, the reader's browser uses the reader's fonts: Cambria Math on Windows, or Latin Modern Math (`fonts-lmodern`) on Debian and Ubuntu. Without one, brackets and radicals do not stretch.

Inline

Source:

```
Mass and energy are related by $E = mc^2$, and $\alpha + \beta = \gamma$
sits in the sentence too.
```

Rendered:

Mass and energy are related by $E = mc^2$, and $\alpha + \beta = \gamma$ sits in the sentence too.

Display

Source:

````
$$
x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}
$$

```math
\int_{-\infty}^{\infty} e^{-x^2}\,dx = \sqrt{\pi}
```
````

Rendered:

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

$$\int_{-\infty}^{\infty} e^{-x^2}\,dx = \sqrt{\pi}$$

Matrices and aligned equations

Source:

````
```math
A = \begin{pmatrix} a & b \\ c & d \end{pmatrix}, \quad \det A = ad - bc
```

```math
\begin{aligned}
\nabla \cdot \mathbf{E} &= \frac{\rho}{\varepsilon_0} \\
\nabla \times \mathbf{B} &= \mu_0 \mathbf{J} + \mu_0 \varepsilon_0
\frac{\partial \mathbf{E}}{\partial t}
\end{aligned}
```
````

Rendered:

$$A = \begin{pmatrix} a & b \\ c & d \end{pmatrix}, \quad \det A = ad - bc$$

$$\begin{aligned}
\nabla \cdot \mathbf{E} &= \frac{\rho}{\varepsilon_0} \\
\nabla \times \mathbf{B} &= \mu_0 \mathbf{J} + \mu_0 \varepsilon_0 \frac{\partial \mathbf{E}}{\partial t}
\end{aligned}$$

Advanced: a third-level sample

This folder (`markdown/advanced/`) holds pages placed at the **third level**. No matter how deeply you split inputs into folders, the sidebar mirrors the folder structure as a tree.

- Even deep hierarchies stay reachable by collapsing/expanding in the sidebar (`sidebar.collapseDepth` lets you "collapse anything deeper than this level by default". It **collapses** rather than **hides**, so every page is always reachable).
- You can link to pages at the same level too: the deep-headings page

Links follow even from deep folders

A link from the third level back up to the first is also converted into a hash route within the single HTML.

- One folder up: Markdown samples
- Root: Overview

Deep headings and the table of contents

The heading depth shown in the in-page table of contents (on the right) can be adjusted with `toc.maxLevel`. By default, **h2–h3** appear in the TOC (h1 is always excluded since it serves as the page title). With `toc.maxLevel: 4`, h4 is included; with `toc.maxLevel: 2`, only h2 appears.

Headings are always shown in the body regardless of whether they appear in the TOC, so making the TOC shallower does not remove information (it's a setting to keep the TOC from getting too long).

Level 2: Overview

This is an h2. It appears in the default TOC (h2–h3).

Level 3: Details

This is an h3. It appears in the default TOC.

Level 4: Notes

This is an h4. It is not in the default TOC, but `toc.maxLevel: 4` makes it appear.

Level 5: Fine print

This is an h5. It appears in the TOC with `toc.maxLevel: 5` or higher.

Level 6: Footnote-like memo

This is an h6, the deepest heading level in Markdown.

Level 2: Summary

On pages that use many deep headings, keeping the TOC shallow improves scannability.

AsciiDoc samples

This sample covers the main syntax from the AsciiDoc Syntax Quick Reference to check its rendering in `monodocs`.

Each item shows the **source (raw AsciiDoc)** first, then the **rendered output (HTML)** below it.

Build example:

```
monodocs build examples/en -o dist/sample.html
```

Open each category from the left sidebar: Lists / Blocks / Tables / Links and images / Wide content.

Section headings

The number of `==` sets the level (`=` is the document title).

Source:

```
=== Level 2

==== Level 3

===== Level 4
```

Rendered:

Level 2

Level 3

Level 4

Paragraphs and line breaks

A blank line separates paragraphs, and a trailing `+` forces a line break. The `[.lead]` role makes a lead paragraph.

Source:

```
This is a paragraph. Blank lines separate paragraphs.

Put a `+` at the end of a line +
to force a line break.

[.lead]
A lead paragraph (the `[.lead]` role).
```

Rendered:

This is a paragraph. Blank lines separate paragraphs.

Put a `+` at the end of a line
to force a line break.

A lead paragraph (the `[.lead]` role).

Text formatting

Decorations such as bold, italic, monospace, highlight, and super/subscript.

Source:

```
* Bold: *bold* and **un**constrained
* Italic: _italic_ and __un__constrained
* Monospace: `monospace`
* Bold italic monospace: `*_combined_*`
* Highlight: #highlighted#
* Super/subscript: E = mc^2^ / H~2~O
* Quotes: "`curly double quotes`" and '`single quotes`'
```

Rendered:

- Bold: **bold** and **un**constrained
- Italic: *italic* and *un*constrained
- Monospace: `monospace`
- Bold italic monospace: ***combined***
- Highlight: highlighted

- Super/subscript: E = mc^2 / H_2O
- Quotes: “curly double quotes” and ‘single quotes’

Replacements (text substitution)

Symbols and arrows are substituted automatically.

Source:

```
(C) (R) (TM) become symbols, arrows -> => <- <= stay as-is, and `...`
becomes an ellipsis.
```

Rendered:

© ® ™ become symbols, arrows → ⇒ ← ⇐ stay as-is, and `…` becomes an ellipsis.

Comments

Lines starting with `//` are comments and are not output.

Source:

```
// This is a comment and is not output.

NOTE: Line comments starting with `//` are not output.
```

Rendered:

> **Note**
> Line comments starting with `//` are not output.

Lists

Each item shows the **source** first, then the **rendered output (HTML)** below it.

Bullet lists (unordered)

The number of `*` sets the nesting.

Source:

```
* Item A
* Item B
** Nested B-1
** Nested B-2
*** Even deeper
* Item C
```

Rendered:

- Item A
- Item B
    - Nested B-1
    - Nested B-2
        - Even deeper
- Item C

Numbered lists (ordered)

`.` makes a numbered list; `..` nests.

Source:

```
. First
. Next
. After that
.. Nested 2-1
.. Nested 2-2
```

Rendered:

1. First
2. Next
3. After that
    a. Nested 2-1
    b. Nested 2-2

Checklists

`[x]` / `[ ]` create checkboxes.

Source:

```
* [x] A completed task
* [ ] An incomplete task
* [*] Completed (alternative notation)
```

Rendered:

- ☑ A completed task
- ☐ An incomplete task
- ☑ Completed (alternative notation)

Description lists

`term

description` creates term/description pairs.

Source:

```
CPU:: Central processing unit.
Memory:: Main memory.
Storage::
  Auxiliary storage. Multi-line descriptions are fine too.
```

Rendered:

CPU

Central processing unit.

Memory

Main memory.

Storage

Auxiliary storage. Multi-line descriptions are fine too.

Terms and nesting

You can nest by changing the level with `::` / `:::`.

Source:

```
Operating system::
  Linux:::
    An open-source OS.
  macOS:::
    Apple's OS.
```

Rendered:

Operating system

Linux

An open-source OS.

macOS

Apple’s OS.

Multiple blocks in a list item

A continuation line (`+`) connects paragraphs or blocks to the same item.

Source:

```
* An item that contains multiple blocks.
+
This is a paragraph of the same item, connected with a continuation line
(`+`).
+
[source,bash]
----
echo "A code block inside a list item"
----
```

Rendered:

- An item that contains multiple blocks.

  This is a paragraph of the same item, connected with a continuation line (`+`).

  ```
  echo "A code block inside a list item"
  ```

Blocks

Each item shows the **source** first, then the **rendered output (HTML)** below it.

Source blocks (syntax highlighting)

A `----` block with `[source,language]` is highlighted by shiki.

Source:

```
[source,python]
----
def fib(n):
    a, b = 0, 1
    for _ in range(n):
        a, b = b, a + b
    return a
----
```

Rendered:

```python
def fib(n):
    a, b = 0, 1
    for _ in range(n):
        a, b = b, a + b
    return a
```

You can annotate with callouts like `// <1>`.

Source:

```
[source,js]
----
function greet(name) { // <1>
  return `Hello, ${name}!`; // <2>
}
----
<1> Define the function.
<2> Return using a template literal.
```

Rendered:

```
function greet(name) { // (1)
  return `Hello, ${name}!`; // (2)
}
```

1. Define the function.
2. Return using a template literal.

Listing blocks / literal blocks

`----` shows text verbatim in monospace; `....` is literal (preserving whitespace).

Source:

```
----
A listing block (monospace, shown as-is)
----

....
A literal block
  Whitespace is preserved.
....
```

Rendered:

```
A listing block (monospace, shown as-is)
```

```
A literal block
  Whitespace is preserved.
```

Example blocks

`====` groups text and other blocks together.

Source:

```
====
An example block. It groups text and other blocks.
====
```

Rendered:

An example block. It groups text and other blocks.

Sidebar blocks

`**` is a box for supplementary information.

Source:

```
****
A sidebar block: a box for supplementary information.
****
```

Rendered:

A sidebar block: a box for supplementary information.

Quote blocks / verse blocks

`[quote]` is a quotation; `[verse]` is a verse block that preserves line breaks and indentation.

Source:

```
[quote, Taro Yamada, The art of single-file distribution]
____
Write docs in pieces; distribute as one file.
____

[verse, Author Name]
____
Line breaks and
  indentation are
preserved in a verse block.
____
```

Rendered:

> Write docs in pieces; distribute as one file.
>
> — Taro Yamada
> *The art of single-file distribution*

```
Line breaks and
  indentation are
preserved in a verse block.
```

— Author Name

Admonitions

Labels such as `NOTE:` create admonitions.

Source:

```
NOTE: A simple NOTE.

TIP: A useful tip.

IMPORTANT: An important note.

WARNING: A warning.

CAUTION: A caution.
```

Rendered:

> **Note**
> A simple NOTE.

> **Tip**
> A useful tip.

> **Important**
> An important note.

> **Warning**
> A warning.

> **Caution**
> A caution.

The `[NOTE]` + `====` block form can contain multiple paragraphs and other blocks.

Source:

```
[NOTE]
====
A block-form NOTE. It can contain multiple paragraphs and other blocks.

* Lists too
* can be included
====
```

Rendered:

> **Note**
> A block-form NOTE. It can contain multiple paragraphs and other blocks.
>
> - Lists too
> - can be included

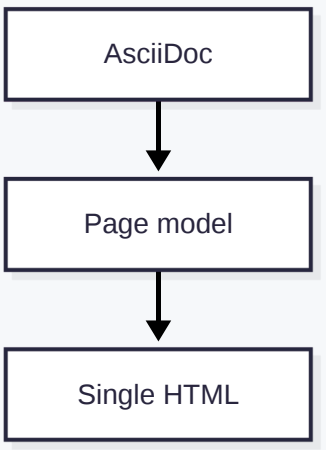
Mermaid (`[source,mermaid]`)

A `[source,mermaid]` block is rendered as a diagram.

Source:

```
[source,mermaid]
----
graph TD
  A[AsciiDoc] --> B[Page model]
  B --> C[Single HTML]
----
```

Rendered:



Passthrough (raw HTML)

Wrapping with `++++` embeds raw HTML as-is (assuming trusted input).

Source:

```
++++
<div style="padding:0.5rem;border:1px dashed #888;border-radius:6px;">
  Raw HTML embedded via passthrough (assuming trusted input).
</div>
++++
```

Rendered:

Raw HTML embedded via passthrough (assuming trusted input).

Tables

Each item shows the **source** first, then the **rendered output (HTML)** below it.

A basic table

`cols` sets column widths and alignment (`^` center / `>` right), `options="header"` makes the first row a header, and `.Title` adds a caption.

Source:

```
.Support status
[cols="1,^1,>1", options="header"]
|===
| Feature | Markdown | AsciiDoc

| Headings
| ✅
| ✅

| Tables
| ✅
| ✅

| Code highlighting
| shiki
| shiki
|===
```

Rendered:

Table 1. Support status

Feature	Markdown	AsciiDoc
Headings	✅	✅
Tables	✅	✅
Code highlighting	shiki	shiki

Cell formatting and multiple lines

You can use `**bold**` and `code` inside cells. The `a|` prefix lets you put AsciiDoc blocks (like lists) in a cell.

Source:

```
[cols="1,2", options="header"]
|===
| Item | Description

| Emphasis
| You can use *bold*, `code`, and _italic_ inside cells.

| Bullets
a|
Use the `a` prefix to put AsciiDoc blocks (like lists) in a cell:

* Item 1
* Item 2
|===
```

Rendered:

Item	Description
Emphasis	You can use **bold**, `code`, and *italic* inside cells.
Bullets	Use the `a` prefix to put AsciiDoc blocks (like lists) in a cell: • Item 1 • Item 2

CSV format

`format=csv` builds a table from comma-separated data.

Source:

```
[%header, format=csv]
|===
Name,Role
core,The core of the conversion
cli,Command line
|===
```

Rendered:

Name	Role
core	The core of the conversion
cli	Command line

Links and images

Each item shows the **source** first, then the **rendered output (HTML)** below it.

Links

URL macros, email, and cross-references (xref) to other files.

Source:

```
* URL macro: {url-repo}[monodocs repository]
* As-is: https://docs.asciidoctor.org/
* Email: mailto:docs@example.com[Documentation team]
* Cross-reference (xref) to another file: xref:index.adoc[Back to top] /
  xref:blocks.adoc[The blocks page]
```

Rendered:

- URL macro: monodocs repository
- As-is: https://docs.asciidoctor.org/
- Email: Documentation team
- Cross-reference (xref) to another file: Back to top / The blocks page

Attributes

An attribute `https://example.com/monodocs` defined in the header can be expanded in the body.

Source:

```
You can expand the attribute `{url-repo}` defined in the header in the
body: {url-repo}
```

Rendered:

You can expand the attribute `https://example.com/monodocs` defined in the header in the body: https://example.com/monodocs

Internal anchors and cross-references

`[#id]` adds an anchor to a heading, and `text` references it. Clicking scrolls to the spot without switching pages.

Source:

```
[#install-section]
=== Installation steps (with an anchor)

This heading has an anchor (ID) via `[#install-section]`.
You can reference it from elsewhere like <<install-section,go to the
installation steps>>.
```

Rendered:

Installation steps (with an anchor)

This heading has an anchor (ID) via `[#install-section]`. You can reference it from elsewhere like go to the installation steps.

Images

`image::` is a block image (with a caption); `image:` is an inline image (images are embedded into the single HTML as data URIs).

Source:

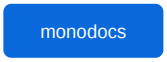
```
.monodocs logo
image::images/logo.svg[Logo, 120]

You can also place an inline image image:images/logo.svg[Logo, 80] within
text.
```

Rendered:



Figure 1. monodocs logo

You can also place an inline image  within text.

UI macros

Macros for key presses, buttons, and menus.

Source:

```
* Key press: kbd:[Ctrl+S] to save, kbd:[Ctrl+Shift+P]
* Button: click btn:[OK]
* Menu: choose menu:File[New > Project]
```

Rendered:

- Key press: kbd:[Ctrl+S] to save, kbd:[Ctrl+Shift+P]
- Button: click btn:[OK]
- Menu: choose menu:File[New > Project]

Footnotes

`[1]` creates a footnote; `[2]` creates a reusable named footnote.

Source:

```
You can attach a footnote to the textfootnote:[This is a footnote. It gets
a unique ID within the single HTML.].
You can also use a named footnotefootnote:ref1[A reusable footnote.] and
reference the same one againfootnote:ref1[].
```

Rendered:

You can attach a footnote to the text[3]. You can also use a named footnote[2] and reference the same one again[2].

---

1. …
2. …
3. This is a footnote. It gets a unique ID within the single HTML.

Wide content

The body is kept at a comfortable reading width (max 860px by default). You can change the fixed width with `html.contentWidth` in the config file, or set it to `full` to span the full width. Set `html.contentWidthToggle` to `false` when readers should not be able to change that width. Set `html.contentWidthDefault` to `wide` to start in wide mode until the reader makes a choice. This sample shows how tables, listings, and images wider than that look in AsciiDoc. Elements that don't fit the body width scroll horizontally within the element, and images are scaled down to fit the body width.

A wide table

A table with many columns won't fit the body width, so you can scroll horizontally within the table to see the remaining columns.

Metric	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
Page views	1200	1840	2010	1750	2230	2680	3120	2990	2640	2880	3310	3720
Unique	820	1190	1320	1180	1460	1710	1980	1920	1700	1840	2100	2350
Bounce (%)	58	55	53	54	51	49	47	48	50	49	46	44

A listing with long lines

A listing block containing long lines scrolls horizontally within the block. Short code is shown as-is.

```
docker run --rm -it --name monodocs-dev -v "$(pwd)":/work -w /work/app -e
NODE_ENV=development -p 4173:4173 monodocs-dev node
packages/cli/dist/index.js serve ../examples/en --host 0.0.0.0 --port 4173
```

```
const width = 860;
```

A large image

An image whose intrinsic width is larger than the body width is scaled down to fit the body width.



Markdown / AsciiDoc mixed

You can mix Markdown and AsciiDoc in the same folder. Cross-format links are also converted into hash routes within the single HTML.

- From this page (Markdown) to the AsciiDoc companion page.
- A link that carries an anchor jumps straight to that heading: “Anchor example” on the companion page.

Anchor target heading

The companion page links to this heading.

Companion (AsciiDoc)

Back to the Markdown page in the same folder: Mixed top.

> **Note**
> AsciiDoc admonitions work as-is too.

Anchor example

This section carries an explicit `[#anchor-demo]` ID, so the Markdown side can jump straight here with `companion.adoc#anchor-demo`.

The other direction works the same: “Anchor target heading” on the mixed top page.

PDF output

`monodocs` can also produce a **PDF via the generated single HTML**. You keep your input split across files while shipping a single deliverable — either HTML or PDF.

```
monodocs build examples/en --format pdf  -o dist/sample.pdf   # PDF only
monodocs build examples/en --format both -o dist/sample       # both HTML
and PDF
```

- `--format both` treats `-o` as a **directory** and writes `docs.html` and `docs.pdf` inside it.
- Internally a headless Chromium opens the single HTML and renders it with the print layout (all pages expanded vertically). **This page itself can be part of the PDF.**

What the PDF includes

- **Bookmarks (outline)**: the same **folder → page** hierarchy as the left sidebar in HTML. Jump to any page from the viewer's bookmarks panel.
- **In-content links**: cross-page links, footnotes, and heading anchors remain clickable inside the PDF.
- **Images**: embedded as data URIs, so the PDF is self-contained (no external files).
- **Code highlighting / diagrams / tables / alerts**: rendered the same as in HTML.
- **Page-break care**: alerts, figures, and code blocks avoid splitting across pages where possible.

> **Note**
>
> The dark/light toggle is an HTML feature. PDFs are produced in the light print color scheme.

Configuration (`monodocs.config.yml`)

```yaml
pdf:
  pageSize: "A4" # A4 / Letter, etc.
  margin:
    top: "20mm"
    right: "15mm"
    bottom: "20mm"
    left: "15mm"
  printBackground: true # print background colors / images
  bookmarks: true # attach bookmarks (sidebar structure)
```

Runtime requirements

> **Important**
>
> PDF output uses a headless Chromium, so **Chromium must be available** in the runtime environment. PDFs are also drawn with the **system fonts of that environment**, so if a font for the characters used in the body (CJK, emoji, …) is missing, they render as tofu: empty or crossed-out boxes. The dev image bundles Japanese (CJK) and emoji fonts.

> **Warning**
>
> The bundled CLI (single `.cjs` / single executable) cannot produce PDFs because it does not ship `puppeteer-core`. Use the package-installed version for PDF output.

Search

The single HTML has full-text search built in. No dictionary and no search runtime are bundled, so the feature costs almost nothing in file size.

Type in the search box in the sidebar and the results narrow over titles, headings, and body text. Every query below matches something that really exists in this sample, so read this page with the box open.

Matching

Matching is case-insensitive and works on substrings, so a query does not have to be a whole word.

Type this	It matches	What it ignores
`INSTALL`	install	letter case
`stall`	install	word boundaries
`ＰＤＦ`	PDF	full-width against half-width

Japanese has more spelling to absorb than English does, and the search folds katakana against hiragana, half-width katakana against full-width, the prolonged sound mark against the dash family, and the wave dash against the full-width tilde. The Japanese sample demonstrates those.

Narrowing with several keywords

Separate keywords with a space and only the pages containing all of them are left.

Type `image` first and several pages line up. Add `embedding` and only the pages containing both are left.

Results that matched a heading

Results are weighted by where the keyword appears: title, then heading, then body. Opening a result that matched a heading goes to that heading rather than to the top of the page.

Type `installation`. It matches a heading in the AsciiDoc sample, and opening it starts you at that heading.

Keyboard

The results can be worked through without leaving the box.

- `↓` and `↑`: move through the results, wrapping at both ends.
- `Enter`: open the selected result, or the top one when you have not moved yet.
- `Escape`: clear the box.

The × button at the right of the box clears it too, along with the results and the marks in the body, and leaves the cursor in the box.

The cursor stays in the search box while you move, so you can keep typing to narrow the list. While an input method is composing, the keys are left to it, so selecting a conversion candidate still works.

Highlighting in the body

Opening a result also marks the keywords in the page you land on. The marks stay while the search is open, so moving to another page does not lose them. Changing or clearing the query puts the body back as it was.

Loose matching

Some differences are matched by loosening the query rather than the text, with no dictionary.

- **English inflections**: a word is matched by its stem from the start of a word. `installing` also finds a page that only says `install`. A word whose stem is shorter than four letters is not loosened.
- **Okurigana variants**: in Japanese, up to two kana between two kanji may be there or not, so `埋込み` finds `埋め込み`. The Japanese sample demonstrates it.

Pages that needed the loose match are ranked after those matched as typed. The marks in the body fall on the text as it is written in the page, not as you typed it.

Generated by monodocs v0.19.0