Getting Started with AsciiDoc

Xintong Wang

Version 1.0, April 21, 2025

Xintong Wang

Table of Contents

Xintong Wang

1. Introduction

1.1. What is AsciiDoc

AsciiDoc is a lightweight markup language[1] used for writing structured documentation, with an `adoc` file extension. It has easy-to-read syntax, rich features, and flexible customization options, making it ideal for creating technical documentation, books, and web content. With Asciidoctor, a Ruby tool, it can be converted into multiple formats such as HTML, PDF, and EPUB.

NOTE

1. *Markup language*: A human-readable language that describes how content should be displayed or structured. Other markup languages include HTML, Markdown, LaTeX, and XML.

1.2. AsciiDoc vs. Markdown

Markdown is one of the most popular markup languages, commonly used in blog posts and is a conventional format for README files in software projects. It is easy to learn, enabling simple and quick content formatting, and has built-in support across various platforms including GitHub, GitLab, and static site generators.

Due to its minimalistic design, it has become a favorite among general users, developers, and bloggers. However, compared to Markdown, AsciiDoc may be a better fit if you want to:

- **Equip your document with more features.**
  AsciiDoc natively supports advanced elements such as merging cells in tables and automatic cross-references between documents, without using extensions or plugins as Markdown does.
- **Customize the content and format more heavily.**
  AsciiDoc allows you to fine-tune the presentation by defining custom roles to apply specific CSS classes, or using attributes to dynamically insert reusable values across your document, which Markdown does not support.
- **Generate output in multiple formats.**
  Using Asciidoctor, you can easily convert an AsciiDoc file to various formats including HTML, PDF, EPUB, DocBook, and slide decks, whereas Markdown requires additional tools or plugins.

2. Setting Up Your Environment

Before you start writing AsciiDoc content, it's important to set up an editor that supports AsciiDoc syntax and features.

1. Install a text editor or integrated development environment (IDE) of your choice.
2. Install the appropriate AsciiDoc plugin or extension to enable syntax highlighting, live preview, and other useful features.

See below for popular editors and their available AsciiDoc support:

Xintong Wang

Copyright © 2025 Xintong Wang

For feedback or inquiries, please contact: xintong3248@gmail.com

Text Editor	AsciiDoc Support
Visual Studio Code (recommended)	Install the `asciidoctor.asciidoctor-vscode` extension.
Atom	Install the `language-asciidoc` and `asciidoc-preview` packages.
IntelliJ IDEA	Install the AsciiDoc Plugin from the JetBrains marketplace.

3. Writing AsciiDoc Documents

With your editor set up, you can begin your AsciiDoc project. To generate polished outputs like PDF or HTML, you'll need supporting files such as YAML configuration and custom font settings, and it's important to organize your project properly.

3.1. Document Structure

Here is an example directory structure:

```
my-project/
    content/
        index.adoc
        chapters/
            introduction.adoc
            setup.adoc
    themes/
        custom-theme.yml
    fonts/
        roboto-regular.ttf
        roboto-bold.ttf
        roboto-italic.ttf
    images/
```

- **AsciiDoc Files (`.adoc`)**

  AsciiDoc files form the main body of your project. You can split a large project into multiple .adoc files and include them using the `include::` directive to keep things modular and manageable. For example, in `my-project/content/index.adoc`:

  ```
  = My Project Title
  Author Name
  :doctype: book
  :toc:
  :sectnums:

  include::chapters/introduction.adoc[]
  include::chapters/setup.adoc[]
  ```

- **YAML Configuration Files (`.yml`)**

  YAML files are often used alongside AsciiDoc when you use conversion tools like Asciidoctor PDF. The YAML theme file allows you to define project-level settings, such as page size, and font settings. You can

Xintong Wang

use a theme that extends from the default AsciiDoc theme:

```
extend: default

page:
  size: A4
  layout: portrait

font:
  catalog:
    NotoSans:
      normal: fonts/NotoSans-Regular.ttf
      bold: fonts/NotoSans-Bold.ttf
      italic: fonts/NotoSans-Italic.ttf
      bold_italic: fonts/NotoSans-BoldItalic.ttf
  base:
    font_family: NotoSans
    font_size: 10

base:
  background_color: #ffffff
  text_color: #333333

table:
  border: 1px solid #ddd
  padding: 0.5em
  background: #fafafa
```

- **Customized Fonts**

  By default, AsciiDoc uses system fonts, but you can specify your own font families for different parts of the document such as body text and code blocks. If your document includes non-English characters, you need to reference a custom font to support these characters. See Google Fonts for a wide selection of fonts.

3.2. Basic Syntax

Create and name your file with the `.adoc` extension. See the following sections for the basics of AsciiDoc syntax and common elements.

3.2.1. Document Header

The document header is *optional*, which encapsulates the document title, author, revision information, document-wide attributes, and other document metadata.

Xintong Wang

Copyright © 2025 Xintong Wang

For feedback or inquiries, please contact: xintong3248@gmail.com

An Example Header	Result

`= Main Title: Subtitle`
`Anonymous Author <hello@email.com>`
`1.1, February 28, 2025`
`:description: The document's description.`
`:doctype: article`
`:toc:`
`:toclevels: 2`
`:sectnums:`
`:sectanchors:`
 	
`== A`
`Document body A.`
 	
`=== Subsection of A`
`Document body in the subsection of A.`
 	
`==== A Further Subsection`
`This section is not shown in Table of Content, because the` `` `:toclevels:` `` `attribute is only set to 2.`
 	
`== B`
`Document body B.`
	Main Title Subtitle
	Anonymous Author – hello@email.com – Version 1.1, February 28, 2025
	Table of Contents
	1. A
	1.1. Subsection of A
	2. B
	1. A
	Document body A.
	1.1. Subsection of A
	Document body in the subsection of A.
	1.1.1. A Further Subsection
	This section is not shown in Table of Content, because the `:toclevels:` attribute is only set to 2.
	2. B
	Document body B.

3.2.2. Sections and Breaks

Aspects	Syntax	Result

Section levels	`== Level 1 Section Title`
	
	`=== Level 2 Section Title`
		1. Level 1 Section Title
		1.1. Level 2 Section Title

Page breaks	`<<<`	

Thematic breaks	3 repeating characters of `'`, `-`, or `*`
	
	`Subject A`
	
	`---`
	
	`Subject B`
		Subject A
		Subject B

Xintong Wang

Aspects	Syntax	Result

Line breaks	`This is the first line. +`
	`This is the second line.`
	
	You can also add `:hardbreaks-option:` under the section title to activate line breaks across the whole section without using "+".
		This is the first line.
		This is the second line.

You can also customize section ID generation, numbers and styles. Refer to the official AsciiDoc documentation here for more details.

3.2.3. Text Format

Aspects	Syntax	Result

Direct formatting:
Enclose the text with symbols.
	`*Bold text*`
	`_Italic text_`
	`` `Monospace` ``
	`Subscript example: H~2~O`
	`Superscript example: x^2^`
		Bold text
		Italic text
		`Monospace`
		Subscript example: H_2O
		Superscript example: x^2

Built-in roles, such as lead, big, small, underline, line-through, and subtitle.
Wrap the text with #.
	`[.big]#Big text#`
	`[.underline]#Underline text#`
	`[.line-through]#Line-through text#`
		Big text
		Underline text
		~~Line-through text~~

Custom roles	Label the text in adoc:
	
	`[.my-custom-role]`
	`A paragraph with customized style.`
	
	`This is [.my-custom-role]#custom text#.`
	
	Define the role in YAML:
	
	`role:`
	`  my-custom-role:`
	`  font-color: #0000FF`
		A paragraph with customized style.
		
		This is custom text.

3.2.4. Lists

Aspects	Syntax	Result
Ordered lists .	`. First item` `. Second item`	1. First item 2. Second item

Unordered lists * or -	`* Item`
	`* Another item`
	`** Nested item`
		• Item
		• Another item
		◦ Nested item

Xintong Wang

Copyright © 2025 Xintong Wang

For feedback or inquiries, please contact: xintong3248@gmail.com

Aspects	Syntax	Result

Checklists `[ ]`	`* [*] Checked`
	`* [x] Also checked`
	`* [ ] Not checked`
		☑ Checked
		☑ Also checked
		☐ Not checked

Description lists `: :`	`Cloud Providers::`
	`  PaaS:::`
	`  . OpenShift`
	`  . CloudBees`
	`  IaaS:::`
	`  . Amazon EC2`
	`  . Rackspace`
		Cloud Providers
		PaaS
		1. OpenShift
		2. CloudBees
		IaaS
		1. Amazon EC2
		2. Rackspace

3.2.5. Links

Aspects	Syntax	Result
URL and Link macros	`Ask questions in the https://chat.asciidoc.org[*AsciiDoc community chat*, window=_blank].`	Ask questions in the **AsciiDoc community chat**.

Cross references	1. Assign an ID to the target section: add `[#set-link-id]` above the target section.
	2. At where you want to reference:
	`See <<set-link-id, Custom Text to Display>> for more details.`
	`Using a section's title such as <<Links>> without assigning ID.`
		See Custom Text to Display for more details.
		Using a section's title such as Links without assigning ID.

Inter-document xref	`Why a tiger icon? See my xref:[path-to-file]#section-id[Text to Display]!`	Why a tiger icon? See my first post!

Xintong Wang

3.2.6. Tables

An Example Table

```
.Table Title
[cols="3,5", options="header"]
[frame="all"]
|===
|Header 1 |Header 2
<|Align to the left  <|Align to the left
>|Align to the right >|Align to the right
^|Center the content ^|Center the content

.3+.^|Merge vertically,
center vertically
|Row 4 column 2
|Row 5 column 2
|Row 6 column 2

2+^|Merge horizontally, center horizontally

a|Using operator `a` for AsciiDoc style:
* List item
* List item
* List item

|Without operator `a`:
* List item
* List item
* List item

h|Using operator `h` for header style
m|Using operator `m` for monospace style
|===
```

Result

Table 1. Table Title

Header 1	Header 2
Align to the left	Align to the left
Align to the right	Align to the right
Center the content	Center the content
Merge vertically, center vertically	Row 4 column 2
	Row 5 column 2
	Row 6 column 2
Merge horizontally, center horizontally	
Using operator `a` for AsciiDoc style: • List item • List item • List item	Without operator `a`: * List item * List item * List item
Using operator `h` for header style	`Using operator m for monospace style`

You can also customize table width, format (such as using a CSV format), table caption, and more. Refer to the official AsciiDoc documentation here for more details.

3.2.7. Images

Syntax

Basic syntax:

```
.Image Caption Displayed
image::<path-to-file>[optional description, width,
height, other definitions]
```

Example:

```
.Tiger Icon
image::/tiger.png[tiger, 50, 50, align="center"]
```

Result

Figure 1. Tiger Icon

Xintong Wang

Copyright © 2025 Xintong Wang

For feedback or inquiries, please contact: xintong3248@gmail.com

3.2.8. Other Useful Elements

Aspects	Syntax	Result

Admonitions, including:
• NOTE
• TIP
• IMPORTANT
• CAUTION
• WARNING
	`WARNING: this is a warning.`
	
	`[NOTE]`
	`====`
	`This is a note section:`
	
	`. List item 1`
	`. List item 2`
	`====`
		WARNING this is a warning.
		
		NOTE This is a note section:
		1. List item 1
		2. List item 2

Footnote (`footnote:[]`)	`This is my footnote footnote:[Example footnote at the end of the page].`	This is my footnote [1].

Source code blocks	`[source, python]`
	`----`
	`first = first-name`
	`last = last-name`
	`print(f"Hello, {first} {last}!")`
	`----`
		`first = first-name`
		`last = last-name`
		`print(f"Hello, {first} {last}!")`

`include` directive: Include content from another file into the current AsciiDoc document.	`include::<path-to-file>[Optional parameters here]`	

`ifdef`/`ifndef` directives: Include or not include content if a specified attribute is set.	`ifdef::env-github[]`
	`So much content here is for GitHub only.`
	`endif::[]`
	
	`ifndef::env-github[]`
	`So much content here is not shown on GitHub.`
	`endif::[]`	

`ifeval` directive: Include the content if the expression evaluates to true.	`ifeval::[{sectnumlevels} == 3]`
	`If the `sectnumlevels` attribute is 3, this sentence is included.`
	`endif::[]`	

4. Converting AsciiDoc

AsciiDoc can be converted into various formats depending on your needs. This section primarily focuses on converting AsciiDoc to PDF, which is a common requirement for documentation and publishing.

4.1. Installing Asciidoctor PDF

Asciidoctor PDF is a Ruby-based tool that converts an AsciiDoc document directly to a PDF document. Here are the steps to install Asciidoctor PDF on different operating systems[1]:

On Windows

If you're using Windows with WSL2 (Windows Subsystem for Linux 2)[2]:

1. Check if you have Ruby available in your WSL2 terminal (e.g., Ubuntu):

```
ruby -v
```

2. If not, install Ruby:

```
sudo apt update
sudo apt install ruby-full
```

3. Install Asciidoctor and Asciidoctor PDF:

```
gem install asciidoctor
gem install asciidoctor-pdf
```

4. Verify the installation of Asciidoctor PDF:

```
asciidoctor-pdf -v
```

NOTE

1. If you are using Linux, the steps are similar to Windows with WSL2.
2. It is recommended to use WSL2 for easier management. If you haven't installed WSL2, refer to the Microsoft's guide for more details. However, if you only need quick and basic PDF export, you can directly install Ruby and Asciidoctor PDF on Windows.

On MacOS

1. Check if you have Ruby available:

```
ruby -v
```

2. If not, install Ruby with Homebrew:

```
brew install ruby
```

3. Add Ruby to your `PATH` by appending the following line to your shell configuration file (e.g., `.zshrc` on macOS):

```
echo 'export PATH="/opt/homebrew/opt/ruby/bin:$PATH"' >> ~/.zshrc
```

4. Install Asciidoctor and Asciidoctor PDF:

```
gem install asciidoctor
gem install asciidoctor-pdf
```

Xintong Wang

Copyright © 2025 Xintong Wang

For feedback or inquiries, please contact: xintong3248@gmail.com

5. Verify the installation of Asciidoctor PDF:

```
asciidoctor-pdf -v
```

4.2. Generating PDF from AsciiDoc

Once you have Asciidoctor PDF installed and your AsciiDoc files ready, you can convert them to PDF using the command line. Simply run `asciidoctor-pdf render-me.adoc` in the terminal.

To customize the output, you can use a custom theme to control layout, fonts, color, table of content, and other elements. Here is a basic command with a customized theme:

```
asciidoctor-pdf --theme path/to/theme/basic-theme.yml -a pdf-
fontsdir="path/to/fonts;GEM_FONTS_DIR" render-me.adoc -o output.pdf
```

Several attributes (`-a`) can be used in the command:

`--theme`
: To apply a custom theme. Without specification, the default theme will be used.

`-a pdf-fontsdir`
: To specify the path to the fonts directory. Include `GEM_FONTS_DIR` if you also use bundled fonts (i.e., the default fonts included with Asciidoctor PDF).

`-o`
: To specify the output file name.

Xintong Wang

[1] Example footnote at the end of the page