Getting started with AsciiDoctor

Table of Contents

Basic markup reference . . . 1
- Headings . . . 1
- Block ID . . . 1
- Paragraph . . . 2
- Inline markup . . . 2
- Lists . . . 3
- Block markup . . . 4
- Links . . . 5
- Complex content . . . 6
- Including content from other files . . . 6

Building content . . . 7

Previewing rendered content . . . 7

As a technical writer who wants produce some documentation, you should be familiar with basic AsciiDoc syntax. The following guide shows of the few most common examples of AsciiDoc syntax usages. For a more in-depth look at AsciiDoc syntax, check out the AsciiDoctor syntax quick reference.

Basic markup reference

Headings

Create headings by stacking = symbols at the beginning of a line. There 5 heading levels, numbered 0 to 4. Your modules should start with a level 0 heading.

```
= This is a level 0 heading
===== This is a level 4 heading
```

Block ID

Attach block IDs to your sections to make internal linking easier. Block IDs must not contain spaces.

```
[id='this-is-a-block-id']
= This is a level 0 heading
```

Paragraph

You don't need to use special markup to type a paragraph in AsciiDoc. You can separate paragraphs by placing an empty line between them.

Like this!

Inline markup

Bold

- **Constrained**

```
*Constrained*
```

- **Un**constrained

```
**Un**constrained
```

Italics

- *Constrained*

```
_Constrained_
```

- *Un*constrained

```
__Un__constrained
```

Monospace

- `Constrained`

```
`Unconstrained`
```

- `Un`constrained

```
``Un``constrained
```

  Use monospace to mark up literal values, such as:

  - file names

- shell commands
- names of services
- variable names and variable values

You can combine inline markup to achieve additional styling options.

Lists

Unordered

An unordered list makes long enumerations:

- clear
- readable
- easy to follow

Ensure that you place an empty line before the first item of an unordered list. Use additional asterisks to create different level nested sub-items.

+

```
This is the list title:
// empty line
* Item 1
** Sub-item 1
* Item 2
* Item 3
```

Ordered

Use an ordered list to describe a sequence of steps:

1. Learn AsciiDoc
2. Write some documentation
3. Publish your documentation
4. Be awesome

```
This is the list title:
// empty line
. First step
.. Sub-step
... Sub-sub-step
. Second step
. Third step
```

Ensure that you place an empty line before the first item of an ordered list. Use additional periods to create different level nested list sub-items.

AsciiDoc also provides markup for other, more specialized list types.

Block markup

Code Block

Use code blocks to show longer sections of code or multi-line commands. Set the `[source]` attribute and specify the language to enable syntax highlighting:

Source

```
[source,bash]
----
#!/bin/bash
file='book.txt'
while read line; do
echo $line
done < $file
----
```

Rendered

```
#!/bin/bash
file='book.txt'
while read line; do
echo $line
done < $file
```

Literal block

Use literal blocks to show terminal output examples or the listed contents of a file:

Source

```
....
$ systemctl status
● hostname.foo
    State: degraded
     Jobs: 0 queued
   Failed: 1 units
    Since: Sun 2019-05-05 20:40:03 CEST; 4h 11min ago
   CGroup: /
           ├─user.slice
           │ └─user-1000.slice
           │   ├─user@1000.service
           │   │ ├─gvfs-goa-volume-monitor.service
           │   │ │ └─2930 /usr/libexec/gvfs-goa-volume-monitor
           │   │ ├─xdg-permission-store.service
....
```

Rendered

```
$ systemctl status
● hostname.foo
    State: degraded
     Jobs: 0 queued
   Failed: 1 units
    Since: Sun 2019-05-05 20:40:03 CEST; 4h 11min ago
   CGroup: /
           ├─user.slice
           │ └─user-1000.slice
           │   ├─user@1000.service
           │   │ ├─gvfs-goa-volume-monitor.service
           │   │ │ └─2930 /usr/libexec/gvfs-goa-volume-monitor
           │   │ ├─xdg-permission-store.service....
...
```

Links

External

Create a link to an external resource:

```
link:https://www.example.com[link text]
```

The following link, for example, takes you to the AsciiDoctor Writer's Guide

Relative

Link to a section in the same document using the section ID:

```
xref:section-id[link text]
```

Complex content

Use the + character to group content of different types together. This ensures that combined content is correctly indented, and preserves the numbering of the steps. The following example features a list with codes examples in two of its steps:

Source

```
. Download `myLatestProject.tar.gz`
. Extract the project files:
+
[source,bash]
----
$ tar -xzvf myLatestProject.tar.gz
----
+
. Install the files:
+
[source,bash]
----
./install.sh
----
```

Rendered

1. Download `myLatestProject.tar.gz`
2. Extract the project files:

   ```
   $ tar -xzvf myLatestProject.tar.gz
   ```

3. Install the files:

   ```
   ./install.sh
   ```

Including content from other files

AsciiDoc allows you to include files in other files, allowing you to compose modules into assemblies and write in a modular way.

Use the following syntax to include a file. Use the `leveloffset` attribute to ensure that the include pieces nest correctly. Specify the location of the included file relative to the one you are included it in. For example:

```
include::modules/using-the-include-directive.adoc[leveloffset=+1]
```

Building content

You should have AsciiDoctor installed on your system. You can use it to build your content.

1. Ensure that you are in the directory that contains your main ADOC file and execute the following command:.

```
$ asciidoctor main_file_name.adoc
```

+ The main file is usually titled `master.adoc`, although you can choose a different name.

Previewing rendered content

To view the built content, open the `main_file_name.html` file in a browser.