AsciiDoc Syntax Quick Reference

Dan Allen

Revision 1.0.0
2013-04-12

About

This guide is a quick reference of the common formatting markup and document elements in the AsciiDoc syntax. AsciiDoc is a mature, lightweight markup language for authoring notes, articles, documentation, books, web pages, slide decks and man pages in plain text.

Note

Several of the examples focus on the output generated by the HTML backend. AsciiDoc produces complementary output when generating DocBook.

Chapter . Paragraphs

Normal

```
Paragraphs don't require any special markup in AsciiDoc.
A paragraph is just one or more lines of consecutive text.

To begin a new paragraph, separate it by at least one blank line.
Consecutive lines of text are joined to form a single paragraph.
```

Result

Paragraphs don't require any special markup in AsciiDoc. A paragraph is just one or more lines of consecutive text.

To begin a new paragraph, separate it by at least one blank line. Consecutive lines of text are joined to form a single paragraph.

Line breaks

```
To make a line break, end the line in a space followed by a plus. +
This results in a visible line break (e.g., <br>) between the lines.

Rubies are red. +
Sapphires are blue.
```

Result

To make a line break, end the line in a space followed by a plus.
This results in a visible line break (e.g., <br>) between the lines.

Rubies are red.
Sapphires are blue.

Literal

```
 A line that begins with one or more spaces is a literal paragraph.
 Literal paragraphs are treated as preformatted text.
 The text is shown in a fixed-width font
```

```
and endlines are preserved.
```

Result

```
A line that begins with one or more spaces is a literal paragraph.
Literal paragraphs are treated as preformatted text.
The text is shown in a fixed-width font
and endlines are preserved.
```

Tip

Use one or two spaces to offset a literal paragraph, but no more. **Be frugal!**

Admonition

```
NOTE: An admonition paragraph draws the reader's attention to
auxiliary information. Its purpose is determined by the label
at the beginning of the paragraph. The admonition is set apart
from the main content and displayed next to its label or
corresponding icon.

Here are the other built-in admonition types:

TIP: Pro tip...

IMPORTANT: Don't forget...

WARNING: Watch out for...

CAUTION: Ensure that...
```

Result

Example

An admonition paragraph draws the reader’s attention to auxiliary information. Its purpose is determined by the label at the beginning of the paragraph. The admonition is set apart from the main content and displayed next to its label or corresponding icon.

Here are the other built-in admonition types:

Example

Pro tip…

Example

Don’t forget…

Example

Watch out for…

Example

Ensure that…

Note

You can also create admonition blocks, which are covered later.

Chapter . Formatted Text

Emphasis

```
_italic phrase_ (or 'italic phrase')

__i__talic le__tt__ers

*bold phrase*

**b**old le**tt**ers

*_bold italic phrase_*

**__b__**old italic le**__tt__**ers
```

Result

italic phrase (or *italic phrase*)

*i*talic le*tt*ers

bold phrase

bold le**tt**ers

bold italic phrase

bold italic le**tt**ers

Monospace

```
+monospace phrase+ and le++tt++ers

+_monospace italic phrase_+ and le++__tt__++ers

+*monospace bold phrase*+ and le++**tt**++ers

+*_monospace bold italic phrase_*+ and le++**__tt__**++ers
```

Result

`monospace phrase` and le`tt`ers

`monospace italic phrase` and le*`tt`*ers

`monospace bold phrase` and le**`tt`**ers

`monospace bold italic phrase` and letters

Custom styling

```
[small]#phrase styled by CSS class .small#

[big]##O##nce upon a time...
```

Result

phrase styled by CSS class .small

Once upon a time…

Superiors and inferiors

```
^superscript phrase^

e = mc^2^

~subscript phrase~

H~2~O
```

Result

superscript phrase

e = mc^{2}

subscript phrase

$H_{2}O$

Quotes

```
`single smart quotes'
```

```
``double smart quotes''
```

Result

‘single smart quotes’

“double smart quotes”

Chapter . Document Header (Optional)

Title only

```
= AsciiDoc Writer's Guide

This guide provides...
```

Tip

The title is the only required element in the document header.

Title and author line

```
= AsciiDoc Writer's Guide
Doc Writer <doc.writer@asciidoc.org>

This guide provides...
```

Title, author line and revision line

```
= AsciiDoc Writer's Guide
Doc Writer <doc.writer@asciidoc.org>
v1.0, 2013-01-01

This guide provides...
```

Important

You cannot have a revision line without an author line.

Document header with attributes

```
= AsciiDoc Writer's Guide
Doc Writer <doc.writer@asciidoc.org>
v1.0, 2013-01-01
:toc:
:imagesdir: assets/images
```

```
:homepage: http://asciidoc.org

This guide provides...
```

Caution

The header may not contain blank lines and must be offset from the content by at least one blank line.

Chapter . Section Titles (Headers)

Article doctype

```
= Document Title (Level 0)

== Section Level 1

=== Section Level 2

==== Section Level 3

===== Section Level 4
```

Warning

When using the article doctype (the default), you can only have one level-0 section title (i.e., the document title) and it must be in the document header.

Note

The number of equal signs matches the heading level in the HTML output. For example, *Section Level 1* becomes an <h2> heading.

Book doctype

```
= Document Title (Level 0)

== Section Level 1

=== Section Level 2

==== Section Level 3

===== Section Level 4

= Section Level 0
```

Important

There are two other ways to define a section title. *Their omission is intentional*. They both require more markup and are therefore unnecessary. The setext[1] title syntax (underlined text) is especially wasteful, hard to remember, hard to maintain and error prone. The reader never sees the extra markup, so why type it? **Be frugal!**

Explicit id

```
[[primitives-nulls]]
== Primitive types and null values
```

Chapter . Include Files

Document parts

```
= Reference Documentation
Lead Developer

This is documentation for project X.

include::basics.adoc[]

include::installation.adoc[]

include::example.adoc[]
```

Common text

```
== About the author

include::author-bio.adoc[]
```

Chapter . Breaks

Horizontal rule

```
'''
```

Result

Page break

```
<<<
```

Chapter . Lists

Unordered, basic

```
- Apples
- Oranges
- Bananas

//

* Apples
* Oranges
* Bananas
```

Result

- Apples
- Oranges
- Bananas

- Apples
- Oranges
- Bananas

Note

Blank lines are required before and after a list and are permitted, but not required, between list items.

Tip

You can force two lists apart with a line comment, as the previous example demonstrates.

Unordered, nested

```
* Apples
** Rome
** Empire
* Oranges
** Navel
** Temple
```

Result

- Apples
  - Rome
  - Empire
- Oranges
  - Navel
  - Temple

Unordered, max nesting

```
* level 1
** level 2
*** level 3
**** level 4
***** level 5
* level 1
```

Result

- level 1
  - level 2
    - level 3
      - level 4
        - level 5

- level 1

Ordered, basic

```
. Step 1
. Step 2
. Step 3
```

Result

1. Step 1
2. Step 2
3. Step 3

Ordered, nested

```
. Step 1
. Step 2
.. Step 2a
.. Step 2b
. Step 3
```

Result

1. Step 1
2. Step 2
    a. Step 2a
    b. Step 2b
3. Step 3

Ordered, max nesting

```
. level 1
```

```
.. level 2
... level 3
.... level 4
..... level 5
. level 1
```

Result

1. level 1

    a. level 2

        i. level 3

            A. level 4

                I. level 5

2. level 1

Labeled, single-line

```
first term:: definition of first term
section term:: definition of second term
```

Result

first term
: definition of first term

section term
: definition of second term

Labeled, multi-line

```
first term::
definition of first term
section term::
definition of second term
```

Result

first term
: definition of first term

section term
: definition of second term

Q&A

```
[qanda]
What is Asciidoctor?::
  An implementation of the AsciiDoc processor in Ruby.
What is the answer to the Ultimate Question?:: 42
```

Result

1. What is Asciidoctor?

   An implementation of the AsciiDoc processor in Ruby.

2. What's the answer to the Ultimate Question?

   42

Mixed

```
Operating Systems::
  Linux:::
    . Fedora
      * Desktop
    . Ubuntu
      * Desktop
      * Server
  BSD:::
    . FreeBSD
    . NetBSD

Cloud Providers::
  PaaS:::
    . OpenShift
```

```
    . CloudBees
  IaaS:::
    . Amazon EC2
    . Rackspace
```

Result

Operating Systems

- Linux
  1. Fedora
     - Desktop
  2. Ubuntu
     - Desktop
     - Server
- BSD
  1. FreeBSD
  2. NetBSD

Cloud Providers

- PaaS
  1. OpenShift
  2. CloudBees
- IaaS
  1. Amazon EC2
  2. Rackspace

Tip

Indenting nested lists and definition list content is optional.

Complex content in outline lists

```
* Every list item has at least one paragraph of content,
  which may be wrapped, even using a hanging indent.
+
Additional paragraphs or blocks are adjoined using an
list continuation on a line adjacent to both blocks.
+
list continuation:: a plus sign (+) on a line by itself

* A literal paragraph does not require a list continuation.

 $ gem install asciidoctor

* AsciiDoc lists may contain any amount of complex content.
+
[cols="2", options="header"]
|===
|Application
|Language

|AsciiDoc
|Python

|Asciidoctor
|Ruby
|===
```

Result

- Every list item has at least one paragraph of content, which may be wrapped, even using a hanging indent.

  Additional paragraphs or blocks are adjoined using an list continuation on a line adjacent to both blocks.

  list continuation
  : a plus sign (+) on a line by itself

- A literal paragraph does not require a list continuation.

```
$ gem install asciidoctor
```

- AsciiDoc lists may contain any amount of complex content, even tables.

Application	Language
AsciiDoc	Python

Application	Language
Asciidoctor	Ruby

Chapter . Links

External

```
http://asciidoc.org - automatic!

http://asciidoc.org[AsciiDoc]

https://github.com/asciidoctor[Asciidoctor @ *GitHub*]
```

Result

http://asciidoc.org - automatic!

AsciiDoc[1]

Asciidoctor @ **GitHub**[2]

Relative

```
link:index.html[Docs]
```

Result

Docs[3]

Cross references

```
See <<paragraphs>> to learn how to write paragraphs.

Learn how to organize the document into <<section_titles,sections>>.
```

Result

See *Paragraphs* to learn how to write paragraphs.

[1] http://asciidoc.org
[2] https://github.com/asciidoctor
[3] index.html

Learn how to organize the document into sections.

Chapter . Images

Block

```
image::sunset.jpg[]

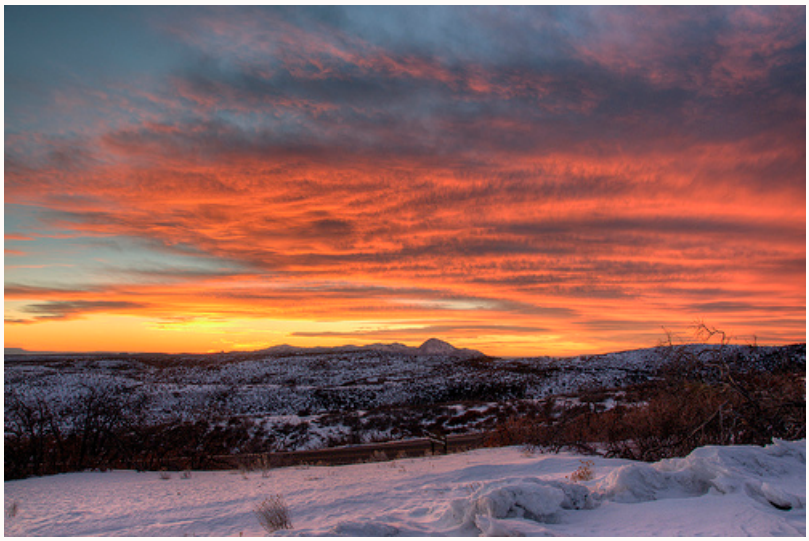
image::sunset.jpg[Sunset]

.A sunset feast!
image::sunset.jpg[Sunset,    300,    200,    link="http://www.flickr.com/photos/
javh/5448336655"]
```

Result

Example

Example



Example

Note

Images resolve relative to the value of the `imagesdir` attribute, defined in the document header or passed as an argument.

Inline

```
Click image:icons/play.png[] to get the party started.

Click image:icons/pause.png[title="Pause"] when you need a break.
```

Result

Click 

to get the party started.

Click 

when you need a break.

Chapter . Source Code

Inline

```
Reference code like +types+ or `methods` inline.
```

Result

Reference code like `types` or `methods` inline.

Literal line

```
 Indent one space to insert a one-liner.
```

Result

```
 Indent one space to insert a one-liner.
```

Literal block

```
....
Copyright (C) 2013 Acme Corporation.

This work is licensed as CC BY-SA, which is the
Creative Commons Attribution 3.0 Unported License.
....
```

Result

```
Copyright (C) 2013 Acme Corporation.

This work is licensed as CC BY-SA, which is the
Creative Commons Attribution 3.0 Unported License.
```

Listing block with title, no syntax highlighting

```
.Gemfile.lock
----
GEM
  remote: https://rubygems.org/
  specs:
    asciidoctor (0.1.1)

PLATFORMS
  ruby

DEPENDENCIES
  asciidoctor (~> 0.1.1)
----
```

Result

Gemfile.lock

```
GEM
  remote: https://rubygems.org/
  specs:
    asciidoctor (0.1.1)

PLATFORMS
  ruby

DEPENDENCIES
  asciidoctor (~> 0.1.1)
```

Code block with title and syntax highlighting

```
[source,ruby]
.app.rb
----
require 'sinatra'

get '/hi' do
  "Hello World!"
end
```

```
----
```

Result

app.rb

```
require 'sinatra'

get '/hi' do
  "Hello World!"
end
```

Code block with callouts

```
[source,ruby]
----
require 'sinatra' <1>

get '/hi' do      <2>
  "Hello World!"  <3>
end
----
<1> Library import
<2> URL mapping
<3> Content for response
```

Result

app.rb

```
require 'sinatra'

get '/hi' do
  "Hello World!"
end
```

1. Library import
2. URL mapping
3. Content for response

Code block sourced from file

```
[source,ruby]
----
include::app.rb[]
----
```

Fenced code block with syntax highlighting

```
```ruby
require 'sinatra'

get '/hi' do
  "Hello World!"
end
```
```

Result

```ruby
require 'sinatra'

get '/hi' do
  "Hello World!"
end
```

Code block without delimiters (no blank lines)

```
[source,xml]
<meta name="viewport"
  content="width=device-width, initial-scale=1.0">
```

Result

```xml
<meta name="viewport"
  content="width=device-width, initial-scale=1.0">
```

Enabling the syntax highlighter

Syntax highlighting is enabled by setting the `source-highlighter` attribute in the document header or passed as an argument.

```
:source-highlighter: pygments
```

The valid options for each implementation are as follows:

AsciiDoc
: pygments, source-highlighter, highlight (default)

Asciidoctor
: coderay, highlightjs (and growing!)

Chapter . More Delimited Blocks

Sidebar

```
.AsciiDoc history
****
AsciiDoc was first released in Nov 2002 by Stuart Rackham.
It was designed from the start to be a shorthand syntax
for producing professional documents like DocBook and LaTeX.
****
```

Result

AsciiDoc history

AsciiDoc was first released in Nov 2002 by Stuart Rackham. It was designed from the start to be a shorthand syntax for producing professional documents like DocBook and LaTeX.

Note

Any block can have a title, positioned above the block. A block title is a line of text that starts with a dot. The dot cannot be followed by a space.

Example

```
.Sample document
====
Here's a sample AsciiDoc document:

[listing]
....
= AsciiDoc Writer's Guide
Dan Allen
:toc:

This guide provides...
....

The document header is useful, but not required.
```

```
====
```

Result

Example

Here's a sample AsciiDoc document:

```
= AsciiDoc Writer's Guide
Dan Allen
:toc:

This guide provides...
```

The document header is useful, but not required.

Admonition

```
[NOTE]
====
An admonition block may contain complex content.

.A list
- one
- two
- three

Another paragraph.
====
```

Result

Example

An admonition block may contain complex content.

A list

- one

- two
- three

Another paragraph.

Blockquote

```
____
A person who never made a mistake never tried anything new.
____

[quote, Albert Einstein]
A person who never made a mistake never tried anything new.

[quote, Abraham Lincoln, Soldiers' National Cemetery Dedication]
____
Four score and seven years ago our fathers brought forth
on this continent a new nation...
____
```

Result

> A person who never made a mistake never tried anything new.

> A person who never made a mistake never tried anything new.
>
> — Albert Einstein

> Four score and seven years ago our fathers brought forth on this continent a new nation…
>
> — Abraham Lincoln *Soldiers' National Cemetery Dedication*

Passthrough

```
++++
<p>
Content in a passthrough block is passed to the output unprocessed.
That means you can include raw HTML, like this embedded Gist:
</p>

<script src="http://gist.github.com/mojavelinux/5333524.js">
```

```
</script>
++++
```

Result

 Content in a passthrough block is passed to the output unprocessed. That means you can include raw HTML, like this embedded Gist:
<script> </script>

Open

```
--
An open block can be an anonymous container,
or it can masquerade as any other block.
--

[source]
--
puts "I'm a source block!"
--
```

Result

An open block can be an anonymous container, or it can masquerade as any other block.

puts "I'm a source block!"

Warning

Asciidoctor doesn't yet support the second use case.

Chapter . Comments

Single line

```
// A single-line comment.
```

Tip

Single-line comments can be used to divide elements, such as two adjacent lists.

Block

```
////
A multi-line comment.

Notice it's a delimited block.
////
```

Chapter . Tables

Table with two rows of content and a header

```
.Applications
[cols="1,1,2" options="header"]
|===
|Name
|Category
|Description

|Firefox
|Browser
|Mozilla Firefox is an open-source web browser.
It's designed for standards compliance,
performance, portability.

|Arquillian
|Testing
|An innovative and highly extensible testing platform.
Empowers developers to easily create real, automated tests.
|===
```

Result

Example

Name	Category	Description
Firefox	Browser	Mozilla Firefox is an open-source web browser. It’s designed for standards compliance, performance, portability.
Arquillian	Testing	An innovative and highly extensible testing platform. Empowers developers to easily create real, automated tests.

Table with an AsciiDoc cell

```
[cols="2,2,5a"]
|===
|Firefox
|Browser
```

```
|Mozilla Firefox is an open-source web browser.

It's designed for:

* standards compliance
* performance
* portability

http://getfirefox.com[Get Firefox]!
|===
```

Result

Firefox	Browser	Mozilla Firefox is an open-source web browser. It’s designed for: • standards compliance • performance • portability Get Firefox[1]!

Table from CSV data

```
[format="csv", options="header"]
|===
Artist,Track,Genre
Baauer,Harlem Shake,Hip Hop
The Lumineers,Ho Hey,Folk Rock
|===
```

Result

Artist	Track	Genre
Baauer	Harlem Shake	Hip Hop
The Lumineers	Ho Hey	Folk Rock

[1] http://getfirefox.com

Table from CSV data in file

```
|===
include::customers.csv[]
|===
```

Table with formatted, aligned and merged cells

```
[cols="e,m,^,>s",width="25%"]
|===
|1 >s|2 |3 |4
^|5 2.2+^.^|6 .3+<.>m|7
^|8
|9 2+>|10
|===
```

Result

1	**2**	3	**4**
5	6		
8			
9		10	7

Chapter . Attributes and Substitutions

Table 2. Text replacements

AsciiDoc	Renders	As Viewed
`(C)`	&#169;	©
`(R)`	&#174;	®
`(TM)`	&#8482;	™
`--`	&#8212;	—
`...`	&#8230;	…
`->`	&#8594;	→
`=>`	&#8658;	⇒
`<-`	&#8592;	←
`<=`	&#8656;	⇐
`Sam's`	Sam’s	Sam’s
`&8364;`	&8364;	€

Tip

Any numbered XML entity reference[1] is supported.

Table 3. Built-in literal attributes

Name	Renders (e.g., `{caret}`)
lt	<
gt	>
amp	&
startsb	[
endsb	]
brvbar	|
caret	^
asterisk	*
tilde	~
apostrophe	'

Name	Renders (e.g., `{caret}`)
backslash	\
two-colons	::
two-semicolons	;;

Table 4. Built-in entity attributes

Name	Renders (e.g., `{nbsp}`)	As Viewed
empty	*blank*	
sp	*single space*	
nbsp		
zwsp	&zwsp;	
wj	&#8288;	
apos	&apos;	'
quot	&quot;	"
lsquo	&#8216;	‘
rsquo	&#8217;	’
ldquo	&#8220;	“
rdquo	&#8221;	”
deg	&#176;	°
plus	&#43;	+

Built-in data attributes

asciidoc (or asciidoctor)
: *blank*

asciidoc-version (or asciidoctor-version)
: Version of the processor

doctype
: Document’s doctype (e.g., article)

backend
: Backend used to render document

localdate
: Local date when rendered

localtime
: Local time when rendered

localdatetime
: Local date & time when rendered

docdate
: Last modified date

doctime
: Last modified time

docdatetime
: Last modified date & time

docfile
: Name of document file

docdir
: Name of document directory

Attribute declaration and usage

```
:homepage: http://asciidoc.org
:homepagelink: http://asciidoctor.org[Asciidoctor]
:desc: AsciiDoc is a mature, plain-text document format for +
       writing notes, articles, documentation, books, and more. +
       It's also a text processor & toolchain for translating +
       documents into various output formats (i.e., backends), +
       including HTML, DocBook, PDF and ePub.
:checkedbox: pass:normal[+[&#10004;]+]

Check out {homepage}[AsciiDoc]!

{desc}

Check out {homepagelink} too!

{checkedbox} That's done!
```

Result

Check out AsciiDoc[2]!

AsciiDoc is a mature, plain-text document format for writing notes, articles, documentation, books, and more. It's also a text processor & toolchain for translating documents into various output formats (i.e., backends), including HTML, DocBook, PDF and ePub.

[2] http://asciidoc.org

Check out Asciidoctor[3] too!

[#] That's done!

[3] http://asciidoctor.org

Chapter . Escaping Text

Backslash

```
\*Stars* is not rendered as bold text.
The asterisks around the word are preserved.

\{author} is not resolved to the author name.
The curly brackets around the word are preserved.

The backslash character is automatically removed.
```

Result

Stars is not rendered as bold text. The asterisks around the word are preserved.

{author} is not resolved to the author name. The curly brackets around the word are preserved.

The backslash character is automatically removed.

Double dollar

```
$$*Stars*$$ is not rendered as bold text.
The asterisks around the word are preserved.

$$&amp;$$ renders as an XML entity instead of &.
```

Result

Stars is not rendered as bold text. The astericks around the word are preserved.

& renders as an XML entity instead of &.

Triple plus and inline passthrough

```
+++<u>underline me</u>+++ renders as underlined text.

pass:[<u>underline me</u>] also renders as underlined text.
```

Result

<u>underline me</u>
renders as underlined text.

<u>underline me</u>
also renders as underlined text.

Backticks

```
`Text in {backticks}` renders exactly as entered, in monospace.
The attribute reference is not resolved.
```

Result

`Text in {backticks}` renders exactly as entered, in monospace. The attribute reference is not resolved.

Chapter . Bibliography

References

```
'The Pragmatic Programmer' <<prag>> should be required reading for
all developers.

[bibliography]
- [[[prag]]] Andy Hunt & Dave Thomas. 'The Pragmatic Programmer:
  From Journeyman to Master'. Addison-Wesley. 1999.
- [[[seam]]] Dan Allen. 'Seam in Action'. Manning Publications.
  2008.
```

Result

The Pragmatic Programmer [prag] should be required reading for all developers.

[prag] Andy Hunt & Dave Thomas. *The Pragmatic Programmer: From Journeyman to Master*. Addison-Wesley. 1999.

[seam] Dan Allen. *Seam in Action*. Manning Publications. 2008.

Chapter . Footnotes

```
A statement footnote:[Clarification about this statement.].

Bold statement footnoteref:[disclaimer,These opinions are my own.].

Another bold statement footenoteref:[disclaimer].
```