

# Living Documentation

# Table of Contents

- Summary ..... 1
- Features ..... 2
  - Cross References ..... 2
    - Scenario: Create a cross reference from an AsciiDoc cell to a section ..... 2
    - Scenario: Create a cross reference using the target section title ..... 3
    - Scenario: Create a cross reference using the target reftext ..... 4
    - Scenario: Create a cross reference using the formatted target title ..... 5
  - Open Blocks ..... 6
    - Scenario: Render an open block that contains a paragraph to HTML ..... 6
    - Scenario: Render an open block that contains a paragraph to DocBook ..... 7
    - Scenario: Render an open block that contains a paragraph to HTML (alt) ..... 8
    - Scenario: Render an open block that contains a paragraph to DocBook (alt) ..... 8
    - Scenario: Render an open block that contains a list to HTML ..... 9
  - Open Blocks ..... 9
    - Scenario: Render a pass block without performing substitutions by default to HTML ..... 9
    - Scenario: Render a pass block without performing substitutions by default to DocBook ..... 10
    - Scenario: Render a pass block performing explicit substitutions to HTML ..... 11
  - Text Formatting ..... 12
    - Scenario: Convert text that contains superscript and subscript characters ..... 12
    - Scenario: Convert text that has ex-inline literal formatting ..... 13
    - Scenario: Convert text that has ex-inline monospaced formatting ..... 14

# Summary

Scenarios			Steps							Features: 4	
Passed	Failed	Total	Passed	Failed	Skipped	Pending	Undefined	Missing	Total	Duration	Status
Cross References											
4	0	4	12	0	0	0	0	0	12	028ms	passed
Open Blocks											
5	0	5	15	0	0	0	0	0	15	043ms	passed
Open Blocks											
3	0	3	9	0	0	0	0	0	9	003ms	passed
Text Formatting											
3	0	3	9	0	0	0	0	0	9	003ms	passed
Totals											
15	0	15	45	0	0	0	0	0	45	079ms	

# Features

## Cross References

In order to create links to other sections

As a writer

I want to be able to use a cross reference macro

**Scenario: Create a cross reference from an AsciiDoc cell to a section**

*Given*

the AsciiDoc source 🍌 (000ms)

```
|===  
a|See <<_install>>  
|===  
  
== Install  
  
Instructions go here.
```

*When*

it is converted to html 🍌 (002ms)

*Then*

the result should match the HTML structure 🍌 (005ms)

```
table.tableblock.frame-all.grid-all.spread  
  colgroup  
    col style='width: 100%;'  
  tbody  
    tr  
      td.tableblock.halign-left.valign-top  
        div  
          .paragraph: p  
            'See  
              a href='#_install' Install  
        .sect1  
          h2#_install Install  
        .sectionbody  
          .paragraph: p Instructions go here.
```

**Scenario: Create a cross reference using the target section title**

*Given*

the AsciiDoc source 📄👍 (000ms)

```
== Section One

content

== Section Two

refer to <<Section One>>
```

*When*

it is converted to html 📄👍 (000ms)

*Then*

the result should match the HTML structure 📄👍 (004ms)

```
.sect1
  h2#_section_one Section One
  .sectionbody: .paragraph: p content
.sect1
  h2#_section_two Section Two
  .sectionbody: .paragraph: p
    'refer to
    a href='#_section_one' Section One
```

**Scenario: Create a cross reference using the target ref text**

*Given*

the AsciiDoc source 🍌 (000ms)

```
[reftext="the first section"]  
== Section One  
  
content  
  
== Section Two  
  
refer to <<the first section>>
```

*When*

it is converted to html 🍌 (000ms)

*Then*

the result should match the HTML structure 🍌 (005ms)

```
.sect1  
  h2#_section_one Section One  
  .sectionbody: .paragraph: p content  
.sect1  
  h2#_section_two Section Two  
  .sectionbody: .paragraph: p  
    'refer to  
    a href='#_section_one' the first section
```

**Scenario: Create a cross reference using the formatted target title**

*Given*

the AsciiDoc source 🍌 (000ms)

```
== Section *One*

content

== Section Two

refer to <<Section *One*>>
```

*When*

it is converted to html 🍌 (001ms)

*Then*

the result should match the HTML structure 🍌 (005ms)

```
.sect1
h2#_section_strong_one_strong
  'Section
  strong One
.sectionbody: .paragraph: p content
.sect1
h2#_section_two Section Two
.sectionbody: .paragraph: p
  'refer to
  a href='#_section_strong_one_strong'
  'Section
  strong One
```

## Open Blocks

In order to group content in a generic container  
As a writer  
I want to be able to wrap content in an open block

**Scenario: Render an open block that contains a paragraph to HTML**



*Given*

the AsciiDoc source 🍌 (000ms)

```
--  
A paragraph in an open block.  
--
```

*When*

it is converted to html 🍌 (008ms)

*Then*

the result should match the HTML source 🍌 (000ms)

```
<div class="openblock">  
<div class="content">  
<div class="paragraph">  
<p>A paragraph in an open block.</p>  
</div>  
</div>  
</div>
```

## Scenario: Render an open block that contains a paragraph to DocBook

*Given*

the AsciiDoc source 🍌 (000ms)

```
--  
A paragraph in an open block.  
--
```

*When*

it is converted to docbook 🍌 (003ms)

*Then*

the result should match the XML source 🍌 (000ms)

```
<simpara>A paragraph in an open block.</simpara>
```

## Scenario: Render an open block that contains a paragraph to HTML (alt)

*Given*

the AsciiDoc source 🍌 (000ms)

```
--  
A paragraph in an open block.  
--
```

*When*

it is converted to html 🍌 (000ms)

*Then*

the result should match the HTML structure 🍌 (019ms)

```
.openblock  
  .content  
    .paragraph  
      p A paragraph in an open block.
```

## Scenario: Render an open block that contains a paragraph to DocBook (alt)

*Given*

the AsciiDoc source 🍌 (000ms)

```
--  
A paragraph in an open block.  
--
```

*When*

it is converted to docbook 🍌 (000ms)

*Then*

the result should match the XML structure 🍌 (003ms)

```
simpara A paragraph in an open block.
```

## Scenario: Render an open block that contains a list to HTML

*Given*

the AsciiDoc source 🍌 (000ms)

```
--  
* one  
* two  
* three  
--
```

*When*

it is converted to html 🍌 (000ms)

*Then*

the result should match the HTML structure 🍌 (004ms)

```
.openblock  
  .content  
    .ulist  
      ul  
        li: p one  
        li: p two  
        li: p three
```

## Open Blocks

In order to pass content through unprocessed

As a writer

I want to be able to mark passthrough content using a pass block

## Scenario: Render a pass block without performing substitutions by default to HTML

*Given*

the AsciiDoc source 🍌 (000ms)

```
:name: value  
  
++++  
<p>{name}</p>  
  
image:tiger.png[]  
++++
```

*When*

it is converted to html 🍌 (000ms)

*Then*

the result should match the HTML source 🍌 (000ms)

```
<p>{name}</p>  
  
image:tiger.png[]
```

**Scenario: Render a pass block without performing substitutions by default to DocBook**

*Given*

the AsciiDoc source 🍌 (000ms)

```
:name: value  
  
++++  
<simpara>{name}</simpara>  
  
image:tiger.png[]  
++++
```

*When*

it is converted to docbook 🍌 (000ms)

*Then*

the result should match the XML source 🍌 (000ms)

```
<simpara>{name}</simpara>  
  
image:tiger.png[]
```

**Scenario: Render a pass block performing explicit substitutions to HTML**

*Given*

the AsciiDoc source 🍌 (000ms)

```
:name: value

[subs="attributes,macros"]
++++
<p>{name}</p>

image:tiger.png[]
++++
```

*When*

it is converted to html 🍌 (000ms)

*Then*

the result should match the HTML source 🍌 (000ms)

```
<p>value</p>

<span class="image"></span>
```

## Text Formatting

In order to apply formatting to the text

As a writer

I want to be able to markup inline text with formatting characters

**Scenario: Convert text that contains superscript and subscript characters**

*Given*

the AsciiDoc source 🍌 (000ms)

```
_v_~rocket~ is the value
^3^He is the isotope
log~4~x^n^ is the expression
M^me^ White is the address
the 10^th^ point has coordinate (x~10~, y~10~)
```

*When*

it is converted to html 🍌 (000ms)

*Then*

the result should match the HTML source 🍌 (000ms)

```
<div class="paragraph">
<p><em>v</em><sub>rocket</sub> is the value
<sup>3</sup>He is the isotope
log<sub>4</sub>x<sup>n</sup> is the expression
M<sup>me</sup> White is the address
the 10<sup>th</sup> point has coordinate (x<sub>10</sub>, y<sub>10</sub>)</p>
</div>
```

**Scenario: Convert text that has ex-inline literal formatting**

*Given*

the AsciiDoc source 🍌 (000ms)

```
Use [x-]`{asciidoctor-version}` to print the version of Asciidoctor.
```

*When*

it is converted to html 🍌 (000ms)

*Then*

the result should match the HTML source 🍌 (000ms)

```
<div class="paragraph">
<p>Use <code>{asciidoctor-version}</code> to print the version of Asciidoctor.</p>
</div>
```

## Scenario: Convert text that has ex-inline monospaced formatting

*Given*

the AsciiDoc source 🍌 (000ms)

```
The document is assumed to be encoded as [x-]+{encoding}+.
```

*When*

it is converted to html 🍌 (000ms)

*Then*

the result should match the HTML source 🍌 (000ms)

```
<div class="paragraph">
<p>The document is assumed to be encoded as <code>UTF-8</code>.</p>
</div>
```