

# Cukedoctor Documentation

Version 1.1.4-SNAPSHOT

# Table of Contents

1. Introduction .....	1
2. Features .....	2
2.1. A feature with background .....	2
2.1.1. Background .....	2
2.1.2. Scenario 1 .....	2
2.1.3. Scenario 2 .....	2
2.2. A feature with output .....	2
2.2.1. Show the current version of sdkman .....	2
2.3. An embed data directly feature .....	3
2.3.1. scenario 1 .....	3
2.3.2. scenario 2 .....	3
2.4. An embed data directly feature .....	3
2.4.1. scenario 1 .....	3
2.4.2. scenario 2 .....	4
2.5. An embed data directly feature .....	4
2.5.1. scenario 1 .....	4
2.5.2. scenario 2 .....	4
2.6. An outline feature .....	5
2.6.1. outline 📄 .....	5
2.7. An outline feature .....	5
2.7.1. outline 📄 .....	5
2.8. Calculator .....	5
2.8.1. Adding numbers .....	5
2.9. Calculator .....	6
2.9.1. Adding numbers .....	6
2.10. Cross References .....	7
2.10.1. Create a cross reference from an AsciiDoc cell to a section .....	7
2.10.2. Create a cross reference using the target section title .....	8
2.10.3. Create a cross reference using the target reftext .....	9
2.10.4. Create a cross reference using the formatted target title .....	10
2.11. Cukedoctor Main .....	11
2.11.1. Generate documentation of a single file .....	12
2.11.2. Generate documentation using multiple files .....	15
2.12. Cukedoctor Main .....	20
2.12.1. Generate documentation of a single file 📄 .....	20
2.12.2. Generate documentation using multiple files 📄 .....	23
2.13. Cukedoctor Main .....	28
2.13.1. Generate documentation of a single file .....	29

2.13.2. Generate documentation using multiple files .....	32
2.14. <b>Discrete class feature</b> .....	37
2.14.1. Render source code .....	37
2.14.2. Render table .....	38
2.15. <b>Do something</b> .....	38
2.15.1. User browses to the site successfully .....	38
2.16. <b>Eat cukes in lot</b> .....	38
2.16.1. Eating many cukes .....	39
2.16.2. Eating many cukes 🐼 .....	39
2.16.3. Eating many cukes 🐼 .....	40
2.16.4. Eating many cukes .....	40
2.17. <b>Eat cukes in lot</b> .....	41
2.17.1. Eating many cukes .....	41
2.17.2. Eating many cukes 🐼 .....	41
2.17.3. Eating many cukes 🐼 .....	42
2.17.4. Eating many cukes .....	42
2.18. <b>Enriched feature</b> .....	43
2.18.1. Scenario with admonition and listing .....	43
2.19. <b>Enriched feature</b> .....	43
2.19.1. Scenario with listing .....	43
2.20. <b>Feature1</b> .....	43
2.20.1. Scenario feature 1 .....	43
2.21. <b>Feature1</b> .....	44
2.21.1. Scenario feature 1 .....	44
2.22. <b>Feature2</b> .....	44
2.22.1. Scenario feature 2 .....	44
2.23. <b>Feature2</b> .....	44
2.23.1. Scenario feature 2 .....	44
2.24. <b>One passing scenario, one failing scenario</b> .....	44
2.24.1. Passing .....	44
2.24.2. Failing 🐼 .....	44
2.25. <b>Open Blocks</b> .....	45
2.25.1. Render an open block that contains a paragraph to HTML .....	45
2.25.2. Render an open block that contains a paragraph to DocBook .....	46
2.25.3. Render an open block that contains a paragraph to HTML (alt) .....	47
2.25.4. Render an open block that contains a paragraph to DocBook (alt) .....	48
2.25.5. Render an open block that contains a list to HTML .....	49
2.26. <b>Open Blocks</b> .....	50
2.26.1. Render a pass block without performing substitutions by default to HTML .....	50
2.26.2. Render a pass block without performing substitutions by default to DocBook .....	51
2.26.3. Render a pass block performing explicit substitutions to HTML .....	52

2.27. <b>Sample test</b> .....	53
2.27.1. Parsing scenarios with multiple examples .....	53
2.27.2. Basic .....	54
2.27.3. Basic failure 🗨 .....	54
2.28. <b>Search</b> .....	54
2.28.1. Find messages by content .....	54
2.29. <b>Text Formatting</b> .....	55
2.29.1. Convert text that contains superscript and subscript characters .....	55
2.29.2. Convert text that has ex-inline literal formatting .....	56
2.29.3. Convert text that has ex-inline monospaced formatting .....	57
2.30. <b>Feature2</b> .....	58
2.30.1. Scenario feature 2 .....	58
2.31. <b>Feature1</b> .....	58
2.31.1. Scenario feature 1 .....	58

# Chapter 1. Introduction

Cukedoctor is a **Living documentation** tool which integrates Cucumber and AsciiDoctor in order to convert your *BDD* tests results into an awesome documentation.

Here are some design principles:

- Living documentation should be readable and highlight your software features;
  - Most bdd tools generate reports and not a truly documentation.
- Cukedoctor **do not** introduce a new API that you need to learn, instead it operates on top of [cucumber json output](#) files;
  - In the 'worst case' to [enhance](#) your documentation you will need to know a bit of [asciidoc markup](#).

In the subsequent chapters you will see a documentation which is generated by the output of [Cukedoctor's BDD tests](#), **a real bdd living documentation**.

# Chapter 2. Features

## 2.1. A feature with background

### 2.1.1. Background

#### Given

this is a background step 🍌 (000ms)

### 2.1.2. Scenario 1

#### Given

this is scenario one step 🍌 (000ms)

### 2.1.3. Scenario 2

#### Given

this is scenario two step 🍌 (000ms)

## 2.2. A feature with output

### 2.2.1. Show the current version of sdkman

### When

I enter "sdk version" 👍 (100ms)

### Then

I see "SDKMAN x.y.z" 👍 (000ms)

Output:

broadcast message  
SDKMAN x.y.z

## 2.3. An embed data directly feature

### 2.3.1. scenario 1

#### Given

I embed data directly 👍 (000ms)

### 2.3.2. scenario 2

#### Given

I embed data directly 👍 (000ms)

#### Given

I embed data directly 👍 (000ms)

## 2.4. An embed data directly feature

### 2.4.1. scenario 1

### Given

I embed data directly 👍 (000ms)

## 2.4.2. scenario 2

### Given

I embed data directly 👍 (000ms)

### Given

I embed data directly 👍 (000ms)

## 2.5. An embed data directly feature

### 2.5.1. scenario 1

### Given

I embed data directly 👍 (000ms)

A paragraph in an open block.

### 2.5.2. scenario 2

### Given

I embed data directly 👍 (000ms)

### Given

I embed data directly 👍 (000ms)



## 2.6. An outline feature

### 2.6.1. outline 🗨️

Table 1. examples1

status
passes
fails

Table 2. examples2

status
passes

## 2.7. An outline feature

### 2.7.1. outline 🗨️

Table 3. examples1

status
passes
fails

Table 4. examples2

status
passes

## 2.8. Calculator

### 2.8.1. Adding numbers

You can **asciidoc markup** in *feature* description.



This is a very important feature!

## Given

I have numbers 1 and 2 🍌 (212ms)



AsciiDoc markup inside **steps** must be surrounded by **curly brackets**.

## When

I sum the numbers using the following java code: 🍌 (001ms)

```
public class Calc {  
    public long sum(int x, int y){  
        return x + y; ①  
    }  
}
```

① This is an asciidoc call inside a feature.



You can use asciidoc in doc strings as well



Steps comments are placed **before** each steps

## Then

I should have 3 as result 🍌 (003ms)

- this is a list of itens inside a feature step
- there is no multiline comment in gherkin
  - second level list item

## 2.9. Calculator

### 2.9.1. Adding numbers

You can use **asciidoc markup** in *feature* description.



This is a very important feature!

### Given

I have numbers 1 and 2 🍌 (114ms)



AsciiDoc markup inside **steps** must be surrounded by **curly brackets**.

### When

I sum the numbers 🍌 (000ms)



Steps comments are placed **before** each steps so this comment is for the **WHEN** step.

### Then

I should have 3 as result 🍌 (001ms)

- this is a list of itens inside a feature step
- there is no multiline comment in gherkin
  - second level list item

## 2.10. Cross References

In order to create links to other sections

As a writer

I want to be able to use a cross reference macro

### 2.10.1. 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.
```

## 2.10.2. 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
```

## 2.10.3. 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
```

### 2.10.4. 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
```

## 2.11. Cukedoctor Main

As a user of CukedoctorMain

I want to generate asciidoc files based on my cucumber test output

So that I can generate wonderful living documentation

### **2.11.1. Generate documentation of a single file**



## Given

A Cucumber json execution file is already generated 🍌 (332ms)

## When

I execute CukedoctorMain with args "-o target/test-classes/outputFile.adoc" "-p target/test-classes/json-output/one\_passing\_one\_failing.json" and "-t Documentation" 🍌 (04s 249ms)

## Then

A file named outputFile.adoc should be generated with the following content: 🍌 (003ms)

```
:toc: right
:backend: html5
:doctype: Documentation
:doctype: book
:icons: font
:!numbered:
:!linkcss:
:sectanchors:
:sectlink:
:docinfo:
:source-highlighter: highlightjs
:toclevels: 3
:hardbreaks:

= *Documentation*

== *Summary*
[col="12^m", options="header,footer"]
|==
3+|Scenarios 7+|Steps 2+|Features: 1

|[[green]]*Passed*#
|[[red]]*Failed*#
|Total
|[[green]]*Passed*#
|[[red]]*Failed*#
|[[purple]]*Skipped*#
|[[maroon]]*Pending*#
|[[yellow]]*Undefined*#
|[[blue]]*Missing*#
|Total
|Duration
|Status

12+^|*<<One-passing-scenario-one-failing-scenario>>*
```

```
|1
|1
|2
|1
|1
|1
|0
|0
|0
|0
|2
|010ms
|[red]##failed*#
12+^|*Totals*
|1|1|2|1|1|0|0|0|0|2 2+|010ms
|===
```

```
== *Features*
```

```
[[One-passing-scenario-one-failing-scenario, One passing scenario, one
failing scenario]]
```

```
=== *One passing scenario, one failing scenario*
```

```
minmax::One-passing-scenario-one-failing-scenario[]
```

```
==== Scenario: Passing
```

```
[small]#tags: @a,@b#
```

## Given

this step passes 👍 (001ms)

```
==== Scenario: Failing
```

```
tags: @a,@c
```

## Given

this step fails 🙄 (008ms)



```
(RuntimeError)
```

```
./features/step_definitions/steps.rb:4:in /^this step fails$/'
```

```
features/one_passing_one_failing.feature:10:in Given this step fails'
```

```
=====
```

### **2.11.2. Generate documentation using multiple files**

## Given

Cucumber multiple json execution files are already generate 🍌 (000ms)

## When

I execute CukedoctorMain with args "-o target/test-classes/outputFile.adoc" "-p target/test-classes/json-output/" and "-t Documentation" 🍌 (01s 135ms)

## Then

A file named outputFile.adoc should be generated with the following content: 🍌 (001ms)

```
:toc: right
:backend: html5
:doctype: Documentation
:doctype: book
:icons: font
:!numbered:
:!linkcss:
:sectanchors:
:sectlink:
:docinfo:
:source-highlighter: highlightjs
:toclevels: 3
:hardbreaks:

= *Documentation*

== *Summary*
[cols="12*^m", options="header,footer"]
|===
3+|Scenarios 7+|Steps 2+|Features: 4

|[[green]]**Passed**#
|[[red]]**Failed**#
|Total
|[[green]]**Passed**#
|[[red]]**Failed**#
|[[purple]]**Skipped**#
|[[maroon]]**Pending**#
|[[yellow]]**Undefined**#
|[[blue]]**Missing**#
|Total
|Duration
|Status

12+^|*<<An-embed-data-directly-feature>>*
```

```
|3
|0
|3
|3
|0
|0
|0
|0
|0
|3
|000ms
|[green]##passed*#
```

```
12+^|*<<An-outline-feature>>*
|0
|0
|0
|0
|0
|0
|0
|0
|0
|0
|0
|000ms
|[green]##passed*#
```

```
12+^|*<<One-passing-scenario-one-failing-scenario>>*
|1
|1
|2
|1
|1
|0
|0
|0
|0
|2
|010ms
|[red]##failed*#
```

```
12+^|*<<Sample-test>>*
|1
|1
|2
|3
|1
|0
|0
|0
|0
```

```
|4
|10s 104ms
|[red]##failed*#
12+^|*Totals*
|5|2|7|7|2|0|0|0|0|9 2+|10s 114ms
|===

== *Features*

[[An-embed-data-directly-feature, An embed data directly feature]]
=== *An embed data directly feature*

minmax::An-embed-data-directly-feature[]
==== Scenario: scenario 1
```

### Given

I embed data directly 👍 (000ms)

```
==== Scenario Outline: scenario 2
```

### Given

I embed data directly 👍 (000ms)

### Given

I embed data directly 👍 (000ms)

=== **An outline feature**

minmax::An-outline-feature[]  
==== Scenario Outline: outline

Table 5. examples1

status
passes
fails

Table 6. examples2

status
passes

=== **One passing scenario, one failing scenario**

minmax::One-passing-scenario-one-failing-scenario[]  
==== Scenario: Passing  
tags: @a,@b

**Given**

this step passes 👍 (001ms)

==== Scenario: Failing  
tags: @a,@c

**Given**

this step fails 🙄 (008ms)



(RuntimeError)  
./features/step\_definitions/steps.rb:4:in /^this step fails\$/'  
features/one\_passing\_one\_failing.feature:10:in Given this step fails'

=== **Sample test**

minmax::Sample-test[]

As a user

I want to do something

In order to achieve another thing

==== Scenario Outline: Parsing scenarios with multiple examples

Table 7. Example

a	b
1	2

==== Scenario: Basic

### Given

I navigate to the home page 👍 (044ms)

### Then

I see the text 'Home' 👍 (001ms)

==== Scenario: Basic failure

### Given

I navigate to the home page 👍 (040ms)

### Then

I see the text 'Hacienda' 🗨️ (10s 017ms)



expected to find text "Hacienda" in "Home | Login Clinical Studies some engaging copy View Available Studies" (RSpec::Expectations::ExpectationNotMetError)  
./features/step\_definitions/study\_admin\_steps.rb:14:in `^I see the text '(.)'\$/'  
features/test\_outline.feature:15:in `Then I see the text 'Hacienda'

=====

## 2.12. Cukedoctor Main

As a user of CukedoctorMain

I want to generate asciidoc files based on my cucumber test output

So that I can generate wonderful living documentation

### 2.12.1. Generate documentation of a single file 🗨️



## Dado

A Cucumber json execution file is already generated 👍 (187ms)

## Quando

I execute CukedoctorMain with args "-o target/test-classes/outputFile.adoc" "-p target/test-classes/json-output/one\_passing\_one\_failing.json" and "-t Documentation" 👍 (17s 662ms)

## Então

A file named outputFile.adoc should be generated with the following content: 🗨️ (002ms)

```
:toc: right
:backend: html5
:doctype: Documentation
:doctype: book
:icons: font
:!numbered:
:!linkcss:
:sectanchors:
:sectlink:
:docinfo:
:source-highlighter: highlightjs
:toclevels: 3
:hardbreaks:

= *Documentation*

== *Summary*
[col="12^m", options="header,footer"]
|==
3+|Scenarios 7+|Steps 2+|Features: 1

|[[green]]*Passed*#
|[[red]]*Failed*#
|Total
|[[green]]*Passed*#
|[[red]]*Failed*#
|[[purple]]*Skipped*#
|[[maroon]]*Pending*#
|[[yellow]]*Undefined*#
|[[blue]]*Missing*#
|Total
|Duration
|Status

12+^|*<<One-passing-scenario-one-failing-scenario>>*
```

```
|1
|1
|2
|1
|1
|1
|0
|0
|0
|0
|2
|010ms
|[red]##*failed*#
12+^|*Totals*
|1|1|2|1|1|0|0|0|0|2 2+|010ms
|===
```

```
== *Features*
```

```
[[One-passing-scenario-one-failing-scenario, One passing scenario, one
failing scenario]]
```

```
=== *One passing scenario, one failing scenario*
```

```
minmax::One-passing-scenario-one-failing-scenario[]
```

```
==== Scenario: Passing
```

```
[small]#tags: @a,@b#
```

## Given

this step passes 👍 (001ms)

```
==== Scenario: Failing
```

```
tags: @a,@c
```

## Given

this step fails 🙏 (008ms)



```
(RuntimeError)
```

```
./features/step_definitions/steps.rb:4:in /^this step fails$/
```

```
features/one_passing_one_failing.feature:10:in Given this step fails'
```

```
IMPORTANT: org.junit.ComparisonFailure:
expected:<...0|0|22+|010ms|====*[Features]*[[One-passing-scena...> but
was:<...0|0|22+|010ms|====*[??title.features??]*[[One-passing-scena...>
    at org.junit.Assert.assertEquals(Assert.java:115)
    at org.junit.Assert.assertEquals(Assert.java:144)
    at
com.github.cukedoctoctor.bdd.CukedoctoctorMainSteps.A_file_named_outputFile_adoc_should
_be_generated_with_the_following_content(CukedoctoctorMainSteps.java:44)
    at .Então A file named outputFile.adoc should be generated with the...

=====
```

### 2.12.2. Generate documentation using multiple files 📄

## Dado

Cucumber multiple json execution files are already generate 👍 (000ms)

## Quando

I execute CukedoctorMain with args "-o target/test-classes/outputFile.adoc" "-p target/test-classes/json-output/" and "-t Documentation" 👍 (01s 194ms)

## Então

A file named outputFile.adoc should be generated with the following content: 🐼 (001ms)

```
:toc: right
:backend: html5
:doctype: Documentation
:doctype: book
:icons: font
:!numbered:
:!linkcss:
:sectanchors:
:sectlink:
:docinfo:
:source-highlighter: highlightjs
:toclevels: 3
:hardbreaks:

= *Documentation*

== *Summary*
[cols="12*^m", options="header,footer"]
|===
3+|Scenarios 7+|Steps 2+|Features: 4

|[[green]]**Passed**#
|[[red]]**Failed**#
|Total
|[[green]]**Passed**#
|[[red]]**Failed**#
|[[purple]]**Skipped**#
|[[maroon]]**Pending**#
|[[yellow]]**Undefined**#
|[[blue]]**Missing**#
|Total
|Duration
|Status

12+^|*<<An-embed-data-directly-feature>>*
```

```
|3
|0
|3
|3
|0
|0
|0
|0
|0
|3
|000ms
|[green]##passed*#
```

```
12+^|*<<An-outline-feature>>*
|0
|0
|0
|0
|0
|0
|0
|0
|0
|0
|0
|000ms
|[green]##passed*#
```

```
12+^|*<<One-passing-scenario-one-failing-scenario>>*
|1
|1
|2
|1
|1
|0
|0
|0
|0
|2
|010ms
|[red]##failed*#
```

```
12+^|*<<Sample-test>>*
|1
|1
|2
|3
|1
|0
|0
|0
|0
```

```

|4
|10s 104ms
|[red]##failed*#
12+^|*Totals*
|5|2|7|7|2|0|0|0|0|9 2+|10s 114ms
|===

== *Features*

[[An-embed-data-directly-feature, An embed data directly feature]]
=== *An embed data directly feature*

minmax::An-embed-data-directly-feature[]
==== Scenario: scenario 1

```

### Given

I embed data directly 👍 (000ms)

```
==== Scenario Outline: scenario 2
```

### Given

I embed data directly 👍 (000ms)

### Given

I embed data directly 👍 (000ms)

=== **An outline feature**

minmax::An-outline-feature[]  
==== Scenario Outline: outline

Table 8. examples1

status
passes
fails

Table 9. examples2

status
passes

=== **One passing scenario, one failing scenario**

minmax::One-passing-scenario-one-failing-scenario[]  
==== Scenario: Passing  
tags: @a,@b

**Given**

this step passes 👍 (001ms)

==== Scenario: Failing  
tags: @a,@c

**Given**

this step fails 🙄 (008ms)



(RuntimeError)  
./features/step\_definitions/steps.rb:4:in /^this step fails\$/'  
features/one\_passing\_one\_failing.feature:10:in Given this step fails'

=== **Sample test**

minmax::Sample-test[]

As a user

I want to do something

In order to achieve another thing

==== Scenario Outline: Parsing scenarios with multiple examples

Table 10. Example

a	b
1	2

==== Scenario: Basic

### Given

I navigate to the home page 🍷 (044ms)

### Then

I see the text 'Home' 🍷 (001ms)

==== Scenario: Basic failure

### Given

I navigate to the home page 🍷 (040ms)

### Then

I see the text 'Hacienda' 🍷 (10s 017ms)



expected to find text "Hacienda" in "Home | Login Clinical Studies some engaging copy View Available Studies" (RSpec::Expectations::ExpectationNotMetError)  
./features/step\_definitions/study\_admin\_steps.rb:14:in `^I see the text '(.)'\$/'  
features/test\_outline.feature:15:in `Then I see the text 'Hacienda'

```
IMPORTANT: org.junit.ComparisonFailure:
expected:<...|92+|10s114ms|====*[Features]*[[An-embed-data-dir...> but
was:<...|92+|10s114ms|====*[??title.features??]*[[An-embed-data-dir...>
  at org.junit.Assert.assertEquals(Assert.java:115)
  at org.junit.Assert.assertEquals(Assert.java:144)
  at
com.github.cukedocter.bdd.CukedocterMainSteps.A_file_named_outputFile_adoc_should
_be_generated_with_the_following_content(CukedocterMainSteps.java:44)
  at .Então A file named outputFile.adoc should be generated with the...

=====
```

## 2.13. Cukedocter Main



As a user of CukedoctorMain

I want to generate asciidoc files based on my cucumber test output

So that I can generate wonderful living documentation

### **2.13.1. Generate documentation of a single file**

## Dado

A Cucumber json execution file is already generated 🍌 (139ms)

## Cuando

I execute CukedoctoMain with args "-o target/test-classes/outputFile.adoc" "-p target/test-classes/json-output/one\_passing\_one\_failing.json" and "-t Documentation" 🍌 (04s 658ms)

## Entonces

A file named outputFile.adoc should be generated with the following content: 🍌 (002ms)

```
:toc: right
:backend: html5
:doctype: Documentation
:doctype: book
:icons: font
:!numbered:
:!linkcss:
:sectanchors:
:sectlink:
:docinfo:
:source-highlighter: highlightjs
:toclevels: 3
:hardbreaks:

= *Documentation*

== *Summary*
[cols="12*^m", options="header,footer"]
|===
3+|Scenarios 7+|Steps 2+|Features: 1

|[[green]]**Passed**
|[[red]]**Failed**
|Total
|[[green]]**Passed**
|[[red]]**Failed**
|[[purple]]**Skipped**
|[[maroon]]**Pending**
|[[yellow]]**Undefined**
|[[blue]]**Missing**
|Total
|Duration
|Status

12+^|*<<One-passing-scenario-one-failing-scenario>>*
```

```
|1
|1
|2
|1
|1
|1
|0
|0
|0
|0
|2
|010ms
|[red]##*failed*#
12+^|*Totals*
|1|1|2|1|1|0|0|0|0|2 2+|010ms
|===
```

```
== *Features*
```

```
[[One-passing-scenario-one-failing-scenario, One passing scenario, one
failing scenario]]
```

```
=== *One passing scenario, one failing scenario*
```

```
minmax::One-passing-scenario-one-failing-scenario[]
```

```
==== Scenario: Passing
```

```
[small]#tags: @a,@b#
```

## Given

this step passes 👍 (001ms)

```
==== Scenario: Failing
```

```
tags: @a,@c
```

## Given

this step fails 🙄 (008ms)



```
(RuntimeError)
```

```
./features/step_definitions/steps.rb:4:in /^this step fails$/
```

```
features/one_passing_one_failing.feature:10:in Given this step fails'
```

```
=====
```

### 2.13.2. Generate documentation using multiple files

## Dado

Cucumber multiple json execution files are already generate 👍 (000ms)

## Cuando

I execute CukedoctorMain with args "-o target/test-classes/outputFile.adoc" "-p target/test-classes/json-output/" and "-t Documentation" 👍 (953ms)

## Entonces

A file named outputFile.adoc should be generated with the following content: 👍 (001ms)

```
:toc: right
:backend: html5
:doctitle: Documentation
:doctype: book
:icons: font
:!numbered:
:!linkcss:
:sectanchors:
:sectlink:
:docinfo:
:source-highlighter: highlightjs
:toclevels: 3
:hardbreaks:

= *Documentation*

== *Summary*
[cols="12*^m", options="header,footer"]
|===
3+|Scenarios 7+|Steps 2+|Features: 4

|[[green]]*Passed*#
|[[red]]*Failed*#
|Total
|[[green]]*Passed*#
|[[red]]*Failed*#
|[[purple]]*Skipped*#
|[[maroon]]*Pending*#
|[[yellow]]*Undefined*#
|[[blue]]*Missing*#
|Total
|Duration
|Status

12+^|*<<An-embed-data-directly-feature>>*
```

```
|3
|0
|3
|3
|0
|0
|0
|0
|0
|0
|3
|000ms
|[green]##passed*#
```

```
12+^|*<<An-outline-feature>>*
|0
|0
|0
|0
|0
|0
|0
|0
|0
|0
|0
|000ms
|[green]##passed*#
```

```
12+^|*<<One-passing-scenario-one-failing-scenario>>*
|1
|1
|2
|1
|1
|0
|0
|0
|0
|0
|2
|010ms
|[red]##failed*#
```

```
12+^|*<<Sample-test>>*
|1
|1
|2
|3
|1
|0
|0
|0
|0
```

```
|4
|10s 104ms
|[red]##failed*#
12+^|*Totals*
|5|2|7|7|2|0|0|0|0|9 2+|10s 114ms
|===

== *Features*

[[An-embed-data-directly-feature, An embed data directly feature]]
=== *An embed data directly feature*

minmax::An-embed-data-directly-feature[]
==== Scenario: scenario 1
```

### Given

I embed data directly 👍 (000ms)

```
==== Scenario Outline: scenario 2
```

### Given

I embed data directly 👍 (000ms)

### Given

I embed data directly 👍 (000ms)

### === An outline feature

```
minmax::An-outline-feature[]  
==== Scenario Outline: outline
```

Table 11. examples1

status
passes
fails

Table 12. examples2

status
passes

### === One passing scenario, one failing scenario

```
minmax::One-passing-scenario-one-failing-scenario[]  
==== Scenario: Passing  
tags: @a,@b
```

#### Given

this step passes 👍 (001ms)

```
==== Scenario: Failing  
tags: @a,@c
```

#### Given

this step fails 🙄 (008ms)



```
(RuntimeError)  
./features/step_definitions/steps.rb:4:in /^this step fails$/'  
features/one_passing_one_failing.feature:10:in Given this step fails'
```

### === Sample test

```
minmax::Sample-test[]
```

As a user

I want to do something

In order to achieve another thing



==== Scenario Outline: Parsing scenarios with multiple examples

Table 13. Example

a	b
1	2

==== Scenario: Basic

### Given

I navigate to the home page 🍷 (044ms)

### Then

I see the text 'Home' 🍷 (001ms)

==== Scenario: Basic failure

### Given

I navigate to the home page 🍷 (040ms)

### Then

I see the text 'Hacienda' 🍷 (10s 017ms)



expected to find text "Hacienda" in "Home | Login Clinical Studies some engaging copy View Available Studies" (RSpec::Expectations::ExpectationNotMetError)  
./features/step\_definitions/study\_admin\_steps.rb:14:in `^I see the text '(.)'\$/  
features/test\_outline.feature:15:in `Then I see the text 'Hacienda'

=====

## 2.14. Discrete class feature

### 2.14.1. Render source code

## Given

the following source code 👍 (267ms)

```
public int sum(int x, int y){  
    int result = x + y;  
    return result; ①  
}
```

① We can have callouts in living documentation&amp;amp;amp;amp;gt;

## 2.14.2. Render table

## Given

the following table 👍 (000ms)

Cell in column 1, row 1	Cell in column 2, row 1
Cell in column 1, row 2	Cell in column 2, row 2
Cell in column 1, row 3	Cell in column 2, row 3

## 2.15. Do something

As a foo...

### 2.15.1. User browses to the site successfully

## Given

User opens a browser 👍 (000ms)

## 2.16. Eat cukes in lot

### 2.16.1. Eating many cukes

#### Given

I have 10 cukes 🍌 (09s 998ms)

#### When

I eat 5 cukes 🍌 (11s 434ms)

#### Then

Am I hungry? "false" 🍌 (18s 585ms)

### 2.16.2. Eating many cukes 🍌

#### Given

I have 0 cukes 🍌 (07s 152ms)

#### When

I eat 0 cukes 🍌 (11s 462ms)

#### Then

Am I hungry? "true" 🍌 (10s 456ms)



```
java.lang.AssertionError: expected:<true> but was:<false>
at org.junit.Assert.fail(Assert.java:88)
at org.junit.Assert.failNotEquals(Assert.java:834)
at org.junit.Assert.assertEquals(Assert.java:118)
at org.junit.Assert.assertEquals(Assert.java:144)
at
com.github.cukedocter.example.EatCukesSteps.amIHungry(EatCukesSte
ps.java:29)
at .Then Am I hungry? "true"(src/test/resources/features/eat-
cukes.feature:7)
```

### 2.16.3. Eating many cukes 🍌

#### Given

I have 2 cukes 🍌 (891ms)

#### When

I eat 3 cukes 🍌 (373ms)

#### Then

Am I hungry? "true" 🍌 (01s 761ms)



```
java.lang.AssertionError: expected:<true> but was:<false>
at org.junit.Assert.fail(Assert.java:88)
at org.junit.Assert.failNotEquals(Assert.java:834)
at org.junit.Assert.assertEquals(Assert.java:118)
at org.junit.Assert.assertEquals(Assert.java:144)
at
com.github.cukedocter.example.EatCukesSteps.amIHungry(EatCukesSte
ps.java:29)
at .Then Am I hungry? "true"(src/test/resources/features/eat-
cukes.feature:7)
```

### 2.16.4. Eating many cukes

#### Given

I have 20600 cukes 🍌 (402ms)

#### When

I eat 20599 cukes 🍌 (159ms)

#### Then

Am I hungry? "false" 🍌 (139ms)

## 2.17. Eat cukes in lot

### 2.17.1. Eating many cukes

#### Given

I have 10 cukes 🍌 (09s 998ms)

#### When

I eat 5 cukes 🍌 (11s 434ms)

#### Then

Am I hungry? "false" 🍌 (18s 585ms)

### 2.17.2. Eating many cukes 🍌

#### Given

I have 0 cukes 🍌 (07s 152ms)

#### When

I eat 0 cukes 🍌 (11s 462ms)

#### Then

Am I hungry? "true" 🍌 (10s 456ms)



```
java.lang.AssertionError: expected:<true> but was:<false>
at org.junit.Assert.fail(Assert.java:88)
at org.junit.Assert.failNotEquals(Assert.java:834)
at org.junit.Assert.assertEquals(Assert.java:118)
at org.junit.Assert.assertEquals(Assert.java:144)
at
com.github.cukedocter.example.EatCukesSteps.amIHungry(EatCukesSte
ps.java:29)
at    .Then Am I hungry? "true"(src/test/resources/features/eat-
cukes.feature:7)
```

### 2.17.3. Eating many cukes 🍌

#### Given

I have 2 cukes 🍌 (891ms)

#### When

I eat 3 cukes 🍌 (373ms)

#### Then

Am I hungry? "true" 🍌 (01s 761ms)



```
java.lang.AssertionError: expected:<true> but was:<false>
at org.junit.Assert.fail(Assert.java:88)
at org.junit.Assert.failNotEquals(Assert.java:834)
at org.junit.Assert.assertEquals(Assert.java:118)
at org.junit.Assert.assertEquals(Assert.java:144)
at
com.github.cukedocter.example.EatCukesSteps.amIHungry(EatCukesSte
ps.java:29)
at .Then Am I hungry? "true"(src/test/resources/features/eat-
cukes.feature:7)
```

### 2.17.4. Eating many cukes

#### Given

I have 20600 cukes 🍌 (402ms)

#### When

I eat 20599 cukes 🍌 (159ms)

#### Then

Am I hungry? "false" 🍌 (139ms)

## 2.18. Enriched feature

### 2.18.1. Scenario with admonition and listing

You can use **asciidoc markup** using feature comments.

#### Given

I have admonition with a listing in feature comments. 🍌 (032ms)



This is a tip with source code inside

```
System.setProperty("INTRO_CHAPTER_DIR", "/home/some/external/folder");
```

## 2.19. Enriched feature

### 2.19.1. Scenario with listing

You can use **asciidoc markup** using feature comments.

#### Given

I have listing in feature comments. 🍌 (000ms)

```
System.setProperty("INTRO_CHAPTER_DIR", "/home/some/external/folder");
```

## 2.20. Feature1

### 2.20.1. Scenario feature 1

#### Given

scenario step 🍌 (647ms)

## 2.21. Feature1

### 2.21.1. Scenario feature 1

#### Given

scenario step 👍 (647ms)

## 2.22. Feature2

### 2.22.1. Scenario feature 2

#### Given

scenario step 👍 (000ms)

## 2.23. Feature2

### 2.23.1. Scenario feature 2

#### Given

scenario step 👍 (000ms)

## 2.24. One passing scenario, one failing scenario

### 2.24.1. Passing

tags: @a,@b

#### Given

this step passes 👍 (001ms)

### 2.24.2. Failing 🙄

tags: @a,@c



## Given

this step fails 🚫 (008ms)



(RuntimeError)

./features/step\_definitions/steps.rb:4:in /^this step fails\$/'  
features/one\_passing\_one\_failing.feature:10:in Given this step fails'

## 2.25. 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

### 2.25.1. 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>
```

## 2.25.2. 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>
```

## 2.25.3. 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.
```

## 2.25.4. 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.
```

### 2.25.5. 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
```

## 2.26. 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

### 2.26.1. 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[]
```

## 2.26.2. 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[]
```

## 2.26.3. 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>
```

## 2.27. Sample test

As a user  
I want to do something  
In order to achieve an important goal

### 2.27.1. Parsing scenarios with multiple examples

scenario with examples

Table 14. examples1

status
passes
fails

Table 15. examples2

status

passes

## 2.27.2. Basic

### Given

I navigate to the home page 🍷 (044ms)

### When

I do something 🍷 (044ms)

### Then

I see the text 'Home' 🍷 (001ms)

## 2.27.3. Basic failure 🍷

### Given

I navigate to the home page 🍷 (040ms)

### Then

I see the text 'Hacienda' 🍷 (10s 017ms)



expected to find text "Hacienda" in "Home | Login Clinical Studies some  
engaging copy View Available Studies"  
(RSpec::Expectations::ExpectationNotMetError)  
./features/step\_definitions/study\_admin\_steps.rb:14:in `^I see the text  
'(.+)'/`  
features/test\_outline.feature:15:in `Then I see the text 'Hacienda'`

## 2.28. Search

### 2.28.1. Find messages by content

tags: @txn

## Dado

a User has posted the following messages: 👍 (111ms)

content
I am making dinner
I just woke up
I am going to work

--

A paragraph in an open block.

--

## 2.29. 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

### 2.29.1. 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>
```

### 2.29.2. 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>
```

## 2.29.3. 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>
```

## 2.30. Feature2

### 2.30.1. Scenario feature 2

#### Given

scenario step 🍌 (000ms)

## 2.31. Feature1

### 2.31.1. Scenario feature 1

#### Given

scenario step 🍌 (313ms)