# Asciidoctor
## manuale d'uso

# INDICE

# 1. COS'è ASCIIDOCTOR

Asciidoctor è un linguaggio di markup che consente di creare documenti in HTML5, pdf, Docbook 5 o 4.5, ed Epub3 in modo sempolice. Per lavorare con Asciidoctor serve ruby, e la relativa gem; serve inoltre un editor di testo semplice. Per produrre file in ofrmato PDF inoltre, è necessario utilizzare **Asciidoctor-pdf**, che estende le funzionalità di Asciidoctor, permettendo la conversione. Quando effettua la conversione di un file dal formato di Asciidoctor, `.adoc` in uno dei vari formati che supporta, Asciidoctor applica un proprio stile di default, personalizzabile creando file in CSS o YAML.

# 2. INSTALLARE ASCIIDOCTOR

Asciidoctor può essere installato tramite il comando `gem`, con il package manager delle varie distribuzioni di linux o via Bundler. L'installazione richiede la presenza di ruby nel sistema; Asciidoctor funziona con diverse versioni di ruby, incluse le API Jruby, Rubinius e Opal.

## 2.1. LINUX E MAC

Ci sono diversi modi di installare Asciidoctor, ma il più semplice è utilizzando il comando `gem` da riga di comando:

```
gem install asciidoctor
```

Un altro modo di installare Asciidoctor è utilizzando i package manager di linux, che variano a seconda della distribuzione:

Debian o Ubuntu

```
apt-get asciidoctor
```

Fedora 21 o precedente

```
sudo yum install asciidoctor
```

Fedora 22 o successivo

```
sudo dnf install asciidoctor
```

una volta installato Asciidoctor, per effettuare il render di documenti in formato PDF, è necessario scaricare Asciidoctor-pdf, che è disponibile come pre-release su rubygems.org. L'installazione è possibile solo da riga di comando:

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

## 2.2. WINDOWS

Per isntallare Asciidoctor su un sistema Windows, bisogna prima installare ruby; esiste un progetto che ne rende l'installazione molto semplice, chiamato Ruby Installer. Una volta scaricato ed eseguita l'installazione, sarà disponibile una command prompt apposita, dalla quale sarà possibile scaricare ed installare la gem di Asciidoctor:

```
gem install asciidoctor
```

Una volta scaricato ed installato Asciidoctor, se si vuole produrre documenti in formato PDF, è necessario installare anche la gem Asciidoctor-pdf, la quale è disponibile solo come pre-release:

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

## 2.3. SOURCE HIGHLIGHTING

Qualora volessimo uno strumento per l'highlighting del codice nel documento, Asciidoctor supporta CodeRay, Rouge e Pygments, i quali sono scaricabili come gem da riga di comando:

```
gem install coderay

gem install rouge

gem install pygments.rb
```

Una volta scaricati uno o più di questi strumenti, possiamo dichiarare nel nostro documento cosa utilizziamo per l'highlighting utilizzando l'attributo `:source-highlighter:` nel documento.

# 3. L'EDITOR

Quando decidiamo di lavorare con Asciidoctor, una decisione importante è la scelta dell'editor di testo da utilizzare: editor di testo complessi come Word o Open office sono sconsigliati, ma allo stesso tempo su piattaforme windows è scosigliato l'uso di trumenti come wordpad o notepad. Questi editor infatti possono complicare la conversione o causare errori nell'interpretazione del file. Gli editor consigliati sono TextMate per Mac

OS X, o GEdit per i sistemi Linux, mentre per Windows è consigliato Notepad++. Alternativamente è possibile utilizzare Vim, Emacs o Sublime Text. Questi editor di testo sono consigliati perché supportano l'highlighting della sintassi di Asciidoctor.

# 4. TIPOLOGIA DEL DOCUMENTO

Asciidoctor permette di scrivere documenti con diverse tipologie, che si differenziano per la struttura. Le tipologie, sono impostate dichiarando un attributo nel documento, o alla conveersione:

Articolo
: la tipologia di default, che può contenere le sezioni abstract, appendice, bibliografia, glossario ed indice, oltre al corpo dell'articolo. L'attributo che definisce l'articolo è `:article:` e può essere omesso, in quanto è la tipologia di default.

Libro
: il libro può contenere più sezioni di livello 0, ed oltre alle sezioni dell'articolo, il libro contene il colofone (note sulla produzione del libro), dedica e prefazione. L'attributo per definire un documento come libro è `:book:`.

Manuale
: il manuale, o MAN page, è utilizzato per produrre documenti simili alla documentazione dei manuali dei sistemi Unix, ha una formattazione particolare, e viene dichiarato con `:manpage:`

# 5. STRUTTURA DEL DOCUMENTO

Un documento Asciidoctor è composto da diversi elementi, suddivisi in elementi block o inline. Ognuno di questi elementi ha una serie di stili, opzioni e funzioni applicabili al loro contenuto; i block element sono elementi che possono occupare più righe, e possono contenere al loro interno altri blocchi. Gli elementi inline invece effettuano delle operazioni su una parte del contenuto di un blocco. Alcuni elementi di tipo block comprendono sezioni, titpoli, header, tabelle e liste, mentre gli elementi inline comprendono quotes, macro o sostituzioni di caratteri.

## 5.1. QUOTES

Con quotes in Asciidoctor si intende una variazione nella formattazione del testo, ad esempio per rendere una parte del testo in grassetto, o in monospace. Ci sono due tipologie di quotes: vincolati (constrained) e liberi (unconstrained).

Per quotes vincolati si intende i quotes che comprendono una o più parole nella loro

interezza, e non compaiono altri caratteri subito prima o subito dopo dei simboli che delimitano i quotes.

Venogno utilizzati con parole singole,

```
Questa macchina è *veloce*
```

con più parole,

```
Questa macchina è *davvero veloce*
```

o quando una parola è seguita da un segno di punteggiatura

```
Non ho mai guidato una macchina *così veloce*!
```

i quotes mostrati nell'esempio rendono il testo che racchiudono in grassetto. Il risultato delle frasi degli esempi è il seguente:

Questa macchina è **veloce**

Questa macchina è **davvero veloce**

Non ho mai guidato una macchina **così veloce**!

I quotes liberi invece servono ad evidenziare parti di una parola o più parole, e vengono usate nei seguenti casi:

- se una lettera, un numero o un underscore precedono o seguono la parte da comprendere nel quote
- se il simbolo di apertura del quote è preceduto da un punto e virgola (;)
- se ci sono degli spazi subito dopo il simbolo di apertura e subito prima il simbolo di chiusura del quote

```
La parola sc**i**enza si scrive con la *i*
```

```
Oggi è il _23_&#8722;__05__&#8722;__2016__
```

```
Ho bisogno di più `` spazio ``
```

La parola sc**i**enza si scrive con la **i**

 

Oggi è il 23-05-2016

Ho bisogno di più `spazio`

Come mostrano gli esempi, i quotes liberi sono delimitati con due simboli invece che uno.

Un caso particolare si presenta se vogliamo alterare una o più parole che sono comprese tra i doppi apici:

```
"` @` "
"` ` @` ` "
"` ` ` @` ` ` "
```

Dato che i doppi apici non sono lettere, numeri o underscore, verrebbe da utilizzare un quote vincolato, ma in questo caso va utilizzato un quote libero. La terza coppia di accenti viene interpretata dal parser di Asciidoctor come parte dei doppi apici. Se effettuassimo un render dell'esempio otterremmo il testo seguente:

```
"@", "@", "@"
```

## 5.2. ATTRIBUTI

Gli attributi sono dichiarazioni effettuate generalmente subito dopo una sezione di livello 0, e che influenzano l'intero documento dalla dichiarazione dell'attributo in poi, tramite comportamenti o stili particolari, come ad esempio la creazione di un indice, o la numerazione delle sezioni del documento. Gli attributi si dividono in 6 categorie, in base alla loro funzione:

- Attributi ambientali (?)
    - Sono attributi che Asciidoctor definisce automaticamente, come la data di creazione del documento, o il percorso del file da convertire. Generalmente sono da considerare attributi di sola lettura, anche se possono essere modificati.
- Attributi integrati
    - Si tratta di attributi definibili ovunque nel documento, ad eccezione di una parte, chiamata attributi dell'header, che vanno definiti all'inizio del documento. Un attributo integrato è visibile e viene applicato solo dopo la sua definizione, e non può essere definito in più punti del documento, se non con il prefisso `@`, ad eccezione dell'attributo `sectnums` che può essere definito più volte nello stesso documento.

- Attributi predefiniti
    - Gli attributi predefiniti vengono utilizzati per sostituire alcuni caratteri se necessario.
- Attributi definiti dall'utente
    - Tutti gli attributi dichiarati e definiti dall'autore; utili per inserire rapidamente contenuto che va utilizzato più volte nel documento.
- API e attributi da riga di comando
    - Attributi appartenenti alle altre categorie ma che possono essere definiti alla conversione, come ad esempio l'attributo ambientale `:backend:` che può essere definito con l'opzione `-b` da riga di comando, o un attributo che definisce la tipologia del documento, definibile con l'opzione `-d` deella riga di comando.
- Attributi degli elementi
    - Attributi definiti in un elemento come una lista o una tabella, i quali hanno validità solo per quell'elemento ed hanno la precedenza sugli attributi definiti nel documento.

### 5.2.1. ASSEGNAZIONE DEGLI ATTRIBUTI

Gli attributi hanno un ordine di interpretazione preciso:

1. Attributi impostati dall'API o dalla riga di comando
2. Attributi impostati nel documento
3. Valore di default degli attributi

È possibile gestire questo ordine in un certo senso: se ad un attributo nell'interfaccia a riga di comando viene aggiunta "`@`" alla fine, la precedenza viene assegnata all'attributo assegnato nel documento, e, qualora non sia presente o assegnato, passa di nuovo alla CLI (command line interface, interfaccia a riga di comando).

Gli attributi vanno definiti con la seguente sintassi:

```
:attributo: valore
```

Come detto in precedenza, gli attributi in Asciidoctor possono richiedere che venga assegnato loro un valore, che può essere numerico, o una stringa, un percorso, un URL o riferimenti ad altri attributi. Inoltre è possibile "disattivare" un attributo impostato in precedenza, inserendo un `!` nell'attributo stesso.

```
:sectnums:
:leveloffset: 3
il valore di leveloffset è {leveloffset}
:!sectnums: :sectnums!:
:imagesdir: ./Immagini
```

Nell'esempio qui sopra vediamo un attributo che non richiede l'inserimento di valori, `:sectnums:` ed un attributo che invece richiede un valore numerico. L'attributo compreso tra parentesi graffe, `{leveloffset}` rappresenta un riferimento al valore dell'attributo `leveloffset`. Nella penultima riga invece, sono riportati i due modi di "disattivare" l'attributo `:sectnums:`; il punto esclamativo per negare l'attributo precedentemente impostato, può essere inserito subito prima o subito dopo il nome dell'attributo stesso, il risultato non cambia. Infine, nell'ultima riga è mostrato un esempio di sintassi che descrive un percorso.

## 5.2.2. SOSTITUIZIONE DEGLI ATTRIBUTI

Una delle feature di Asciidoctor è quella di poter utilizzare sostituzioni di caratteri come i caratteri speciali; queste sostituzioni sono disponibili anche negli attributi, e possono essere utilizzatio per creare del contenuto da richiamare più volte nel documento utilizzando solo il riferimento all'attributo, così da non digitarne il contenuto; le sostituzioni verranno viste più nel dettaglio in seguito, ma per ora vediamo un esempio:

```
:app-name: pass:quotes[MyApp^(C)^]
```

Nell'esempio riportato qui sopra, la macro pass applica la sostituzione, e se dovessimo fare riferimento all'attributo app-name, otterremmo questo risultato: MyApp©

## 5.2.3. ATTRIBUTI SU RIGHE MULTIPLE

In certi casi, come ad esempio la creazione di un attributo definito dall'utente per inserire automaticamente nel documento elementi lunghi come paragrafi interi o righe di codice, può essere utile dividere il contenuto dell'attributo in più righe in modod da renderlo facilmente leggibile da chi andrà a vedere il documento in formato .adoc. Un attributo del genere è definito come ogni altro attributo, ed ogni riga termina con una backslash (`\`).

```
:attributo-lungo: questo è un attributo lungo, è talmente lungo che \
per facilitare la lettura del contenuto di questo attributo molto lungo \
a chi dovesse vedere il documento non renderizzato, \
quindi il documento in formato originale, è stato diviso in più righe, \
altrimenti la sua lettura potrebbe risultare difficile.
```

## 5.2.4. LIMITI DEGLI ATTRIBUTI

Gli attributi di Asciidoctor, seppur molto utili e versatili, hanno delle limitazioni riguardo al loro contenuto; e certi elementi non sono supportati all'interno dell'attributo stesso.

**Cos'è supportato**:

- contenuto semplice
    - un numero, una stringa, un percorso o un URL
- riferimenti ad altri attributi
- formattazione testuale
    - testo in **grassetto**, *corsivo* o `monospace` e sostituzione testuale
- macro

**Cosa non è supportato**:

- liste
- paragrafi multipli
- tipologie di markup che necessitano di whitespace

## 5.2.5. ATTRIBUTI DEGLI ELEMENTI

È possibile assegnare ad un elemento inline o block, oppure una macro, uno o più attributi, e questo si ottiene attraverso l'uso di liste di attributi, le quali hanno la precedenza sugli attributi impostati nel documento per l'elemento specifico a cui fanno riferimento. Una lista di attributi è un insieme di attributi specifici, separati tra loro da una virgola, e compresi tra delle parentesi quadre:

```
[positional-attribute, positional-attribute, named-attribute="valore"]
```

**Positional attribute**: il positional attribute in un elemento inline, viene chiamato *role*, mentre in una macro e un elemento di tipo block come una tabella o un paragrafo è chiamato *style*.

**Named attribute** i named attribute sono attributi a cui viene assegnato, tramite l'uso di un `=` un valore compreso tra doppi apici. Un esempio di named attribute è l'attributo `cols` che indica il numero di colonne di una tabella. Per rendere un named attribute indefinito, se in precedenza era stato definito, basta assegnargli il valore `none`.

 

### ROLE

Il role è utilizzato principalmente per l'output HTML. L'attributo role infatti, una volta effettuato il render in HTML, diventa la classe di un elemento. Per dichiarare un role ci sono 3 modi: il primo è quello di precedere il nome del role da assegnare con un `.`, il secondo è quello di utilizzare il named attribute `role`, ed il terzo, che è valido solo per gli elementi inline è quello di inserirlo per primo nella lista degli attributi di quell'elemento. Come la classe in HTML, anche il role può contenere più valori:

```
[.role1.role2.role3]<elemento generico>
[role="role1, role2, role3"]<elemento generico>
[role]<elemento inline>
[.role1.role2.role3]<elemento inline>
```

### STYLE

Lo style viene utilizzato per cambiare l'aspetto o il comportamento di un intero elemento di tipo block o macro. In una lista di attributi, è il primo elemento se la lista fa riferimento ad un block o ad una macro. Ad un paragrafo ad esempio può essere assegnato l'attributo `source` per fare in modo che l'intero paragrafo venga renderizzato come un blocco di codice (come è stato fatto per tutti gli esempi di questo manuale).

### ID

L'id di un elemento ha come scopo principale quello di fornire un'"ancora" per la creazione di cross reference, e nel caso l'output sia HTML, viene inserito come id dell'elemento. Oltre a questa funzione però l'id permette l'applicazione di uno stile particolare ad un elemento. L'id di un elemento è definito con un `#`, compreso come il role tra parentesi quadre. possiamo inoltre definire assieme l'id di un elemento ed il suo role:

```
[#id.role]<elemento>
```

## 5.2.6. ATTRIBUTI MANCANTI

Se viene fatto un riferimento ad un attributo che non è stato definito, Asciidoctor generalmente non mostra la riga che contiene quell'attributo; tuttavia, per evidenziare questi problemi, nelle ultime release, sono stati inseriti due attributi nuovi: *attribute-missing e attribute-undefined*, che permettono all'utente di specificare il comportamento che deve seguire asciidoctor quando incontra attributi mancanti o non definiti.

## ATTRIBUTE-MISSING

Questo attributo viene utilizzato per definire il comportamento di Asciidoctor quando viene fatto un riferimento ad un attributo non esistente. L'attributo accetta 4 possibili valori: `skip`, `drop`, `drop-line` e `warn`.

- `skip`
    - l'impostazione di default, il riferimento viene mostrato così come è stato scritto;
- `drop`
    - il riferimento viene rimosso;
- `drop-line`
    - l'intera riga contenete il riferimento viene rimossa;
- `warn`
    - viene mostrato un messaggio di avviso che il riferimento manca;

| Valore | Risultato |
|---|---|
| skip | Ciao, {nome}! |
| drop | Ciao, ! |
| drop-line | |
| warn | WARNING: skipping reference to missing attribute: name |

## 5.2.7. ATTRIBUTE UNDEFINED

L'attributo attribute-undefined controlla come vengono gestiti gli statement che disattivano un attributo:

```
{set:name!}
```

Le due opzioni disponibili sono `drop` e `drop-line`. Come con l'attributo attribute-missing, `drop` sostituisce lo statement con una stringa vuota, mentre `drop-line` rimuove la riga che lo contiene. L'impostazione di default è `drop-line`, e quindi è consigliato mettere questi statement in una riga a parte.

## 5.3. HEADER

L'header di un documento contiene il titolo del documento, il sottotitolo, infomrazioni sull'autore e sulla versione del documento, e tutti gli attributi che vanno applicati all'intero documento, inclusi gli attributi definiti dall'utente. L'headere non è necessario su un documento di tipo `article` o `book`, mentre è obbligatorio nel `manpage`. Un header deve sempre iniziare con il titolo, seguito da due righe opzionali che contengono i dati dell'autore e la versione del documento. Subito dopo vanno inseriti tutti gli attributi che si vogliono applicare a tutto il documento. La fine dell'header è delimitata dalla prima riga vuota incontrata dopo il titolo; quindi un header non può contenere righe vuote, ma può contenere commenti.

### 5.3.1. TITOLO

Il titolo del documento è scritto come una sezione di livello 0, dichiarata con il simbolo uguale seguito da almeno uno spazio, e di seguito il testo del titolo.

```
= Il Ristorante Al Termine Dell'Universo

Il succo della storia fin qui.
Al principio fu creato l'Universo. Questo fatto ha sconcertato non poche
persone ed è stato considerato dai più come una cattiva mossa.
```


*Figura 1. Un titolo con paragrafo*

I documenti di tipo `article` o `manpage` possono avere solo una sezione di livello 0, mentre un documento di tipo `book` può avere diverse sezioni di livello 0. Se il documento è di tipo `book`, la prima sezione di livello 0 rappresenta il titolo del documento, mentre le successive sezioni rappresentano il titolo delle parti del libro. Alternativamente ad una sezione di livello 0, il titolo del documento può essere dichiarato con l'attributo `:doctitle:`.

Il sottotitolo del documento è definito tramite l'utilizzo dei due punti (`:`) seguiti da uno spazio; nel caso il titolo sia composto da più elementi di punteggiatura di questo tipo, solo il contenuto dopo gli ultimi due punti viene interpretato come sottotitolo del documento. Inoltre il sottotitolo non viene interpretato se il formato di output è HTML5.

```
= Guida Galattica Per Autostoppisti: Il Ristorante Al Termine
Dell'Universo
```

In questo caso il titolo sarà "Guida Galattica Per Autostoppisti", mentre il sottotitolo è "Il Ristorante Al Termine Dell'Universo".

```
= Guida Galattica Per Autostoppisti: Parte 2: Il Ristorante Al Termine
Dell'Universo
```

Nell'esempio qui sopra invece, il titolo del documento sarà "Guida Galattica Per Autostoppisti: Parte 2" mentre il sottotitolo è, come sopra "Il Ristorante Al Termine Dell'Universo".

C'è la possibilità di utilizzare un speratore diverso dai due punti per delimitare l'inizio di un sottotitolo, tramite l'utilizzo dell'attributo `title-separator`

## 5.3.2. AUTORE E CONTATTI

A seguito del titolo e del sottotitolo, Asciidoctor fornisce la possibilità di inserire l'autore del documento, ed eventualmente un contatto mail, o un'URL. Questi dati vengono inseriti in una nuova riga sotto il titolo, ed il contatto mail o URL va compreso tra partentesi angolari (< e >).

```
= Il Ristorante Al Termine Dell'Universo
Douglas Noel Adams, <http://douglasadams.com[douglasadams.com]>

== Sull'autore

{firstname} {middlename} {lastname}, (1952-2001) è stato un autore e
sceneggiatore britannico.
```


*Figura 2. Titolo e autore*

Asciidoctor interpreta il contenuto nella sezione riguardante l'autore, e associa automaticamente il contenuto ai seguenti attributi:

- `firstname`
    - Il nome dell'autore.
- `middlename`
    - Il secondo nome dell'autore.
- `lastname`
    - Il cognome dell'autore.
- `author`
    - Il nome completo dell'autore.
- `authorinitials`
    - Le iniziali dell'autore (nome, secondo nome, cognome).
- `email`
    - L'indirizzo email o l'URL inserita dopo il nome.

Gli attributi vengono completati automaticamente in base alla posizione in cui sono stati inseriti. Se ad esempio il nome dell'autore viene inserito nell'ordine inverso, cioò cognome e nome, allora il cognome dell'autore risulterà nell'attributo `firstname` ed il nome nell'attributo `lastname`; l'attributo `middlename` viene compilato se il nome comprende più di due elementi. Questi attributi sono inoltre impostabili dall'utente nell'header.

I formati di output `html` e `docbook` possono accettare più autori di un documento. Ogni autore va definito sulla stessa riga, e vanno separati tra loro con un punto e virgola; il primo autore avrà gli attributi elencati sopra, mentre gli autori successivi saranno assegnati ad attributi simili, il cui nome termina con un underscore seguito dalla posizione dell'autore nell'elenco, ad esempio `author_2`, `author_3` e così via.

### 5.3.3. VERSIONE, DATA E NOTE

La versione di un documento contiene 3 attributi:

- `revnumber`:
    - indica la versione del documento, la quale deve contenere almeno un carattere numerico. Ogni lettera o simbolo che precedono il numero non vengono mostrati. Se l'attributo `revdate` non viene impostato, `revnumber` deve finire con una virgola, o iniziare con la lettera "v" , ad esempio `v0.82a`.

- `revdate`:
    - indica la data del documento, e se non viene specificata, viene utilizzato l'attributo `docdate`.
- `revremark`:
    - l'attributo `revremark` permette di inserire un breve commento riguardo alla versione del documento.

### 5.3.4. METADATI

Nel formato di output `html` è possibile aggiungere dei metadati al documento, tramite lutilizzo di attributi particolari; i più comuni sono `description` e `keywords`

**description**

consente di inserire una descrizione del documento in un tag `meta`.

**keywords**

permette di inserire una lista di parole chiave separate da virgola in un tag meta.

### 5.3.5. PREAMBOLO

Il contenuto compreso tra l'header di un documento e la prima sezione di livello 1, o 0 se si tratta di un output di tipo `book` viene interpretata da Asciidoctor come preambolo di un testo, ed è opzionale

## 5.4. SEZIONI

le sezioni dividono il contenuto di un documento Asciidoctor in base ad un sistema gerarchico, e sono definite con dei titoli della sezione.

```
= Sezione di livello 0

== Sezione di livello 1

=== Sezione di livello 2

==== Sezione di livello 3

===== Sezione di livello 4

====== Sezione di livello 5
```


*Figura 3. Titoli delle sezioni*

I titoli della sezione seguono delle regole precise sulla loro posizione: come detto in precedenza, un documento non può avere più di una sezione di livello 0 a meno che non venga impostato il formato di output `book`, e le sezioni devono essere inserite in ordine in base al livello:

```
= Titolo

= sezione di livello 0 illegale

== sezione di livello 1

==== sezione di livello 3 illegale
```

I livelli delle sezioni vanno inseriti in ordine, ovvero una sezione di livello 1 va seguita da una sezione di livello 2, la quale non può contenere sezioni di livello 1 e così via. Asciidoctor supporta, oltre ai titoli delle sezioni definiti con il simbolo `=`, anche la definizione con il simbolo `#`, propria del linguaggio Markup.

### 5.4.1. ID

Alle sezioni viene assegnato un id automaticamente, in base al loro titolo, utilizzabile per cross-reference. Gli id generati in questo modo sono composti così: `-titolo_sezione`; ogni id inizia con un `-`, e gli spazi sono separati da underscore. Per elmininare il prefisso dell'id, o per modificarlo, va assegnato un valore all'attributo `idprefix`; per togliere il prefisso automatico basta non assegnare un valore all'attributo, semplicemente dichiarandolo.

Possiamo inoltre inserire manualmente degli id che puntano al titolo della sezione, utilizzando una lista di elementi separati da virgole racchiusi in due coppie di parentesi quadre:

```
[[sezione 1, capitolo 1, cose]]
== Capitolo 1
```

Nell'esempio riportato sopra vengono definiti 3 id per il `Capitolo 1`, al quale possiamo adesso fare riferimento con le parole chiave `sezione 1`, `capitolo 1` e `cose`.

## 5.4.2. NUMERAZIONE

Asciidoctor permette tramite l'utilizzo di alcuni attributi, la possibilità di numerare le sezioni automaticamente; tramite l'utilizzo dell'attributo `sectnums`.

È possibile, se la numerazione delle sezioni è attiva, disattivarla per non numerare alcune sezioni. Per fare ciò basta alternare la negazione dell'attributo `sectnums!`, e l'attributo `sectnums`.

```
:sectnums:

= Sezione numerata

== Sezione numerata

:sectnums!:

=== Sezione non numerata

=== Sezione non numerata

:sectnums:

==== Sezione numerata
```

Come mostrato nell'esempio, i le sezioni compresi tra la negazione dell'attributo e la nuova dichiarazione dell'attributo, non sono numerate. Un ulteriore possibilità per la numerazione delle sezioni, sta nel poter definire la "profondità" della numerazione, ovvero quanti livelli vengono numerati. Di default la numerazione avviene per tutte le sezioni fino al livello 3, ovveero tutte le sezioni dal livello 1 al 3. La profondità della numerazione è stabilita tramite l'attributo `sectnumlevels`, definibile solo nell'header.

## 5.4.3. STILI DELLE SEZIONI

Le sezioni, se il formato di output è `article` o `book` possono avere degli stili predefiniti, tipici di tesi, articoli di giornale, o libri. Questi stili sono definiti generalmente

all'inizio di un blocco di testo o subito prima di una sezione di livello 1, e sono definiti specificando lo stile tra parentesi quadre. Gli stili possibili sono:

- abstract
- appendix
- bibliography
- colophon
- dedication
- glossary
- index
- part-introduction
- preface

Lo stile `part-introduction` è disponibile solo nel caso di un formato di output di tipo `book`

```
[abstract]
== Titolo

contenuto della sezione
```

Nell'esempio qui sopra, l'intera sezione utilizzera lo stile dell'abstract.

## 5.5. BLOCCHI

In Asciidoctor i blocchi sono paragrafi, liste o elementi di un documento che assolvono a funzioni specifiche; un esempio di blocco utilizzato finora è il blocco di tipo `source` che mostra il testo all'interno in monospace e non interpretando il contenuto al suo interno.

### 5.5.1. TITOLI

I blocchi possono avere un titolo, che va assegnato prima dell'inizio del blocco:

*Blocco con titolo*

```
.Blocco con titolo
[source, Asciidoctor]
----
Contenuto del blocco
----
```

 

L'esempio riportato qui sopra mostra un blocco di tipo `source`, con il titolo, ed al suo interno la sintassi per definire il titolo del blocco.

## 5.5.2. BLOCCHI DELIMITATI

Con blocco delimitato si intende un blocco il cui inizio e fine sono dichiarati dall'utente attraverso dei segni di punteggiatura particolari; dentro questi marcatori possono essere contenute righe vuote. Il blocco delimitato non viene chiuso finché il marcatore che ne indica la chiusura non viene trovato. Inoltre il contenuto di un blocco viene interpretato in modi differenti a seconda del tipo di blocco delimitato. Le varie tipologie vengono definite dai marcatori utilizzati.

| Tipo di blocco | Nome del blocco | Marcatore | Scopo |
|---|---|---|---|
| Admonition | [etichetta] | Qualsiasi marcatore | Contenuto a cui viene associato un tag o un'icona |
| Comment | Nessuno | //// | Testo che non viene processato durante il render |
| Example | [example] | ==== | Definisce un blocco admonition oppure un esempio |
| Fenced | Nessuno | | Il contenuto viene mostrato così come è scritto, senza interpretazione |
| Listing | [listing] | --- | Il contenuto viene mostrato così come è scritto, senza interpretazione |
| Literal | [literal] | .... | Il contenuto viene mostrato così come è scritto, senza interpretazione |
| Open | Quasi tutti i nomi degli altri blocchi | – | Blocco generico che può essere utilizzato al posto degli altri blocci, ad eccezione del passthrough e della tabella |
| Passthrough | [pass] | ++++ | Il contenuto viene mostrato così come è scritto, senza interpretazione |

| Tipo di blocco | Nome del blocco | Marcatore | Scopo |
|---|---|---|---|
| Quote | [quote] | ____ | Una citazione, con la possibilità di inserirne l'autore |
| Sidebar | [sidebar] | **** | Testo e contenuto renderizzato a lato del testo del documento |
| Source | [source] | ---- | Il contenuto vien mostrato così come è scritto, senza interpretazione |
| Stem | [stem] | ++++ | Contenuto che non viene processato ma viene inviato direttamente ad uno strumento di interpretazione come AsciiMath o LaTeX math |
| Table | Nessuno | \|=== | Mostra il contenuto sottoforma di tabella |
| Verse | [verse] | ____ | Un verso con la possibilità di inserirne l'autore |

## 5.6. PARAGRAFI

Il paragrafo è l'elemento dove solitamente si trova la maggior parte del contenuto di un documento. Per questo motivo, Asciidoctor non richiede alcun tipo di markup per delimitare un paragrafo. Ogni paragrafo infatti inizia e termina con una riga vuota; se nello scrivere si va a capo e si inserisce del testo su una nuova riga, quella riga viene comunque considerata parte del paragrafo.

```
Il contenuto di questa riga
e di questa, fa parte dello stesso paragrafo.

Il paragrafo precedente termina con una riga vuota,
quindi il contenuto di queste righe
fa parte di un nuovo paragrafo.
```

Anche se viene premuto invio e viene inserito del testo nella riga immediatamente successiva a quella in cui si stava scrivendo, il contenuto risulta nello sesso paragrafo, e quando viene effettuato il render, viene interpretato come se fosse stato scritto sulla stessa riga. Per fare in modo che il contenuto inserito nella nuova riga vada

effettivamente nella nuova riga, viene inserito un + al termine della riga stessa:

```
Nel mezzo del cammin di nostra vita +
mi ritrovai per una selva oscura, +
ché la diritta via era smarrita.
```

Il contenuto, una volta interpretato da Asciidoctor, verrà messo su più righe, così come è stato scritto. Alternativamente è possibile utilizzare un attributo dichiarato nell'header, per far si che la formattazione delle righe venga rispettata, o come opzione nel blocco per forzare la formattazione solo nel blocco; L'attributo è `hardbreaks`, e l'assegnazione dell'attributo al blocco viene effettuata con l'attributo options: `options="hardbreaks"`, oppure nella sua versione abbreviata `%hardbreaks`.

# 5.7. FORMATTAZIONE DEL TESTO

Come visto in precedeza con i quotes, è possibile alterare il testo per ottenere diversi effetti. Questa variazione nella formattazione è ottenuta comprendendo il testo da modificare in marcatori, chiamati quotes. Nelle ultime versioni di Asciidoctor, la formattazione del testo ha iniziato a separarsi dai quotes, con l'aggiunta di funzionalità. È possibile ottenere i seguenti tipi di formattazione testuale:

- grassetto
- corsivo
- virgolette ed apostrofi curvi
- apice e pedice
- monospace
- evidenziata

## 5.7.1. GRASSETTO E CORSIVO

Come visto in precedenza è possibile enfatizzare il testo trasformandolo in grassetto o in corsivo, o entrambi. Per rendere del testo in grassetto basta inserire un asterisco (*) all'inizio ed alla fine del testo da trasformare, oppure nel caso siano necessari i quotes liberi, due asterischi. Allo stesso modo il testo è trasformato in corsivo inserendo all'inizio ed alla fine del contenuto da modificare, un'underscore (_) o due, a seconda dei casi.

## 5.7.2. VIRGOLETTE ED APOSTROFI CURVI

Di default Asciidoctor non renderizza i doppi apici, o virgolette (") e gli apici, o apostrofi (') come curvi; è possibile però, utilizzando l'accento grave (` ) è possibile trasformarli in apici singoli e doppi curvi:

```
"`Il tempo è un'`illusione. L'`ora di pranzo è una doppia illusione`"
```

Il contenuto riportato nell'esempio, una volta interpretato da Asciidoctor verrà renderizzato così:

> "Il tempo è un'illusione. L'ora di pranzo è una doppia illusione"

## 5.7.3. APICE E PEDICE

Un'altra possibilità che Asciidoctor offre è quella di avere del testo come apice o pedice, ovvero del testo spostato verso l'alto o verso il basso rispetto alla riga, comuni nella scrittura di espressioni matematiche o formule chimiche. Per trasformare del testo in apice o pedice basta comprendere il contenuto tra due accenti circonflessi (`^`), mentre per avere un testo sottoforma di pedice, si usa la tilde (`~`).

```
Se x = a^y^, allora y = log~a~ x.
```

> Se $x = a^y$, allora $y = \log_a x$.

## 5.7.4. MONOSPACE

Il testo in monospace è testo che viene renderizzato come se fosse all'interno di un blocco di tipo source. In genere viene utilizzato per fare riferimento ad elementi propri di un linguaggio, come è stato fatto finora per gli attributi di Asciidoctor. Per avere del testo in monospace basta comprenderlo tra una coppia o due di accenti gravi (` ).

```
Ha stampate in copertina, a grandi caratteri che ispirano fiducia, le
parole `NON FATEVI PRENDERE DAL PANICO`
```

## 5.7.5. EVIDENZIATO

Se il formato di output è HTML, è possibile ottenere del testo evidenziato, comprendendolo tra una coppia di cancelletti (`#`). Nei formati come il pdf, il testo evidenziato viene mostrato come testo normale, ma i cancelletti non vengono renderizzati. Nell'HTML finale, i marcatori sdel testo evidenziato saranno sostituiti dai tag `<marked>`.

```
"Quarantadue!" urlò Loonquawl. "Questo è tutto ciò che sai dire dopo un
lavoro di #sette milioni e mezzo di anni?#"
```

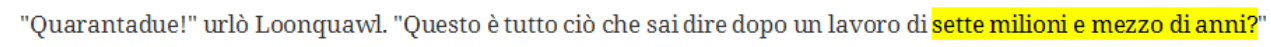


*Figura 4. Testo evidenziato*

### 5.7.6. ORDINE DEI MARCATORI

Le varie tipologie di formattazione del testo possono essere combinate tra loro, ma va fatto disponendo i marcatori in una sorta di ordine gerarchico, riportato di seguito, dai primi marcatori, ovvero quelli più esterni, verso l'interno:

1. Marcatori che agiscono sullo sfondo: I primi marcatori da inserire sono il cancelletto (#) o l'accento grave (` ). Non è possibile avere del testo in monospace evidenziato e viceversa.
2. Apice e pedice
3. Grassetto
4. Corsivo
5. Virgolette ed apostrofi curvi

Se queste regole non vengono rispettate, il marcatore più esterno applica la trasformazione, mentre quelli interni vengno interpertati come elementi di testo.

## 5.8. ELENCHI

Asciidoctor permette la creazione di diverse tipologie di elenchi: ordinati, checklist, labeled list, elenchi puntati, e con diversi tipi di "punteggiatura". Per definire un elenco basta inserire in righe consecutive i vari elementi, preceduti da dei marcatori che definiscono il tipo di elenco stesso. Essendo dei blocchi, gli elenchi possono avere dei titoli.

### 5.8.1. ELENCHI PUNTATI

Un elenco puntato è un elenco i cui elementi sono preceduti da un punto, o da un'altro simbolo, che non sia un numero. Un elemento di un elenco puntato è dichiarato con un asterisco (*) o un meno (-), seguito da uno spazio, ed il contenuto dell'elemento dell'elenco. Inoltre ogni elemento può avere dei sotto elenchi, fino a 5 livelli di profondità.

```
* elemento 1
** elemento 1-1
** elemento 1-2
*** elemento 1-2-1
* elemento 2
```

- elemento 1
    - elemento 1-1
    - elemento 1-2
        - elemento 1-2-1
- elemento 2

Un elenco puntato può avere diversi tipi di simboli prima degli elementi; i simboli disponibili, assieme alla loro parola chiave sono:

- quadrato, `square`
- cerchio (l'opzione di default), `circle`
- disco, `disc`
- nessuno, ma con indentazione, `none` o `no-bullet`
- senza indentazione, né simbolo (solo per HTML), `unstyled`

La dichiarazione del simbolo va effettuata prima dell'elemento dell'elenco, tra parentesi quadre:

```
[square]
- elemento 1
- elemento 2
- elemento 3
-- elemento 3-1
--- elemento 3-2
```

- elemento 1
- elemento 2
- elemento 3
- elemento 3-1
- elemento 3-2

Se un elemento di un elenco comprende più righe, è possibile includere quelle righe inserendo un segno più (+) tra una riga e l'altra:

```
- elemento 1
+
elemento 1 continuato
- elemento 2
```

- elemento 1

  elemento 1 continuato

- elemento 2

## 5.8.2. CHECKLIST

Asciidoctor permette la creazione di checklist, ovvero elenchi nei quali gli elementi sono preceduti da riquadri che possono contenere una spunta. Per definire un elemento di una checklist viene utilizzato come marcatore il segno `-`, seguito da due possibili marcatori, separati da uno spazio:

- Due parentesi quadre, aperta e chiusa, separate da uno spazio (`[ ]`). Questo indica un riquadro non spuntato.
- Due parentesi quadre, aperta e chiusa, contenenti un asterisco o una x (`[*]` o `[x]`).

```
- [ ] elemento non spuntato
- [x] elemento spuntato
- elemento senza checkbox
```

elemento non spuntato

elemento spuntato

elemento senza checkbox

Inoltre, se l'output è HTML, le checkbox dei vari elementi possono essere rese interattive, permettendo all'utente di spuntarle. Questo è ottenuto attraverso l'opzione `interactive`.

## 5.8.3. ELENCHI ORDINATI

I tipi di elenchi visti finora sono semplici liste di elementi. Oltre a questa tipologia, è possibile creare elenchi ordinati, i cui elementi sono numerati. Ci sono inoltre diversi tipi di numerazione degli elementi di un elenco ordinato:

| Tipo | Nome | Scopo |
|---|---|---|
| Araba | arabic | L'elemento è preceduto da un numero arabo. |
| Decimale | decimal | L'elemento è preceduto da un numero arabo, ma i numeri a una cifra vengono preceduti da uno 0. I numeri da 1 a 9 diventano quindi numeri da 01 a 09. |
| Minuscola | loweralpha | L'elemento è preceduto da una lettera minuscola. |
| Maiuscola | upperalpha | L'elemento è preceduto da una lettera maiuscola. |
| Minuscola Romana | lowerroman | L'elemento è preceduto da un numero romano, scritto in lettere minuscole (`i`, `xiii`, `iv`). |
| Maiuscola Romana | upperroman | L'elemento è preceduto da un numero romano, scritto in lettere maiuscole (`I`, `XIII`, `IV`). |
| Minuscola Greca | lowergreek | L'elemento è preceduto da una lettera minuscola dell'alfabeto greco. |

Per dichiarare un elemento di un elenco come elemento ordinato viene utilizzato come marcatore il punto (`.`), mentre per definire il tipo di numerazione, come per il simbolo negli elenchi puntati, viene inserito il nome della numerazione tra parentesi quadre. Inoltre se un elemento contiene a sua volta un elenco numerato, Asciidoctor utilizza per il sotto-elenco, e per tutti gli altri sotto-elenchi di quel livello un sistema di numerazione diverso.

**NOTE** Le tipologie di numerazione minuscola greca e decimale sono disponibili solo per l'output HTML

Un'altra possibilità che ci offre asciidoctor è quella dell'attributo `start`, che permette all'utente di controllare il punto di partenza della numerazione, ovvero da che numero o lettera contare gli elementi;

```
[start=3]
. elemento 1
. elemento 2
. elemento 3
```

3. elemento 1
4. elemento 2
5. elemento 3

## 5.8.4. LABELED LIST

Un altro modo di elencare le cose in Asciidoctor è quello di utilizzare le labeled list: si tratta di elenchi in cui ogni elemento può essere seguito da del testo indentato. Questo tipo di elenco viene dichiarato con l'uso di due due punti (`::`).

```
CPU:: Il cervello del computer
Hard Disk:: Spazio di archiviazione del sistema operativo e dei file
RAM:: Memoria temporanea per le operazioni della CPU
```

CPU
: Il cervello del computer

Hard Disk
: Spazio di archiviazione del sistema operativo e dei file

RAM
: Memoria temporanea per le operazioni della CPU

Il contenuto di una labeled list può essere un qualsiasi blocco di Asciidoctor. Questo tipo di elenco quindi può contenere altri elenchi, paragrafi o tabelle.

Un altro impiego per la labeled list è la creazione di un elenco di domande e risposte, tramite l'opzione `qanda`. La sintassi per definire questo tipo di labeled list è la stessa, ma il testo della label deve terminare con un punto di domanda.

```
[qanda]
Qual è la risposta alla domanda fondamentale sulla vita, sull'universo e
tutto quanto?::
42
```

Qual è la risposta alla domanda fondamentale sulla vita, sull'universo e tutto quanto?

42

## 5.9. TABELLE

Le tabelle sono tra gli elementi più complessi disponibili su Asciidoctor, ma rimangono comunque semplici da leggere anche nel formato originale, ed intuitive da scrivere. Una tabella, come gli altri blocchi in Asciidoctor, utilizza dei marcatori per delimitarne l'inizio e la fine. Ogni tabella è aperta e chiusa da `|===`, il contenuto è diviso in righe e colonne, che contengono celle. Ogni cella inizia con una barra verticale (`|`), ed ogni riga della tabella generalmente deve avere lo stesso numero di colonne.

```
|===
|riga 1 colonna 1   |riga 1 colonna 2   |riga 1 colonna 3
|riga 2 colonna 1   |riga 2 colonna 2   |riga 2 colonna 3
|===
```

La tabella mostrata qui sopra è una semplice tabella a 2 righe e 3 colonne. Le celle sono state messe in ordine per renderne più semplice la lettura, ma utilizzando l'attributo `cols`, è possibile dichiarare tutte le righe e le colonne di una tabella in una riga. L'attributo infatti permette di dichiarare quante colonne avrà la tabella, e quindi organizzare le celle dichiarate in righe in base al numero di colonne.

```
[cols="3"]
|===
|riga 1 colonna 1|riga 1 colonna 2|riga 1 colonna 3|riga 2 colonna 1|riga
2 colonna 2|riga 2 colonna 3
|===
```

Un altro metodo per definire una tabella è quello di definire una cella per riga:

```
[cols="3"]
|===
|riga 1 colonna 1
|riga 1 colonna 2
|riga 1 colonna 3
|riga 2 colonna 1
|riga 2 colonna 2
|riga 2 colonna 3
|===
```

I 3 esempi mostrati qui sopra, una volta interpretati danno lo stesso risultato:

| riga 1 colonna 1 | riga 1 colonna 2 | riga 1 colonna 3 |
|---|---|---|
| riga 2 colonna 1 | riga 2 colonna 2 | riga 2 colonna 3 |

## 5.9.1. COLONNE

Come appena visto, il numero di colonne di una tabella è definito dal numero di celle definite nella prima riga non vuota dopo il marcatore di inizio della tabella, o con l'attributo `cols`. Se all'attributo viene assegnato solo un numero, quel numero determina quante colonne avrà la tabella, e creerà una tabella con quel numero di colonne, tutte della stessa larghezza. Oltre ad un singolo numero però, l'attributo `cols` accetta anche un elenco di elementi separati da virgola.

```
[cols="1,1,1,1"]
|===
|riga 1 colonna 1|riga 1 colonna 2|riga 1 colonna 3|riga 1 colonna 4
|===
```

Nell'esempio qui sopra, la tabella avrà 4 colonne della stessa larghezza.

### FORMATTAZIONE DELLE COLONNE

L'attributo `cols`, oltre a definire il numero di colonne, permette di definire anche la formattazione delle colonne stesse, attraverso l'uso di marcatori che definiscono stile, allineamento e larghezza della colonna, più un marcatore chiamato *moltiplicatore*.

Il moltiplicatore (`*`) viene utilizzato quando si vuole applicare una tipologia di formattazione a tutte le colonne. Se viene utilizzato, va messo sempre prima del marcatore. Nell'esempio seguente, il moltiplicatore assicura che le impostazioni di default vengano applicate a tutte le colonne.

```
[cols="3*"]
```

L'allineamento del contenuto delle celle viene impostato con 3 marcatori, che determinano se il contenuto è centrato (`^`), allineato a sinistra (`<`), o allineato a destra (`>`). Ad esempio per centrare il contenuto di tutte le celle di una tabella, verrà usato `^` subito dopo il moltiplicatore:

```
[cols="3*^"]
|===
|riga 1 colonna 1 |riga 1 colonna 2 |riga 1 colonna 3
|riga 2 colonna 1 |riga 2 colonna 2 |riga 2 colonna 3
|===
```

| riga 1 colonna 1 | riga 1 colonna 2 | riga 1 colonna 3 |
|---|---|---|
| riga 2 colonna 1 | riga 2 colonna 2 | riga 2 colonna 3 |

Se però fosse necessario cambiare l'allineamento del contenuto di una sola colonna, è possibile farlo sempre utilizzando il moltiplicatore: supponiamo che nella tabella di 3 colonne vista finora, il contenuto dell'ultima colonna debba essere allineato a destra; per farlo dobbiamo assegnare le impostazioni di default alle prime due colonne, ed aggiungere una colonna con l'allineamento desiderato:

```
[cols="2*,^"]
|===
|riga 1 colonna 1 |riga 1 colonna 2 |riga 1 colonna 3
|riga 2 colonna 1 |riga 2 colonna 2 |riga 2 colonna 3
|===
```

Se effettuiamo un render della tabella mostrata qui sopra, vediamo come il contenuto della colonna 3 sia allineato a destra:

| riga 1 colonna 1 | riga 1 colonna 2 | riga 1 colonna 3 |
|---|---|---|
| riga 2 colonna 1 | riga 2 colonna 2 | riga 2 colonna 3 |

Allo stesso modo, per applicare un allineamento diverso ad ogni singola colonna, viene usata una lista di elementi separati da virgola che ne definiscono l'allineamento.

```
[cols="<,2*^,>"]
```

L'attributo `cols` definisce una tabella a 4 colonne dove il contenuto della prima è allineato a sinistra, la seconda e terza colonna hanno il contenuto centrato, e la quarta colonna lo avrà allineato a destra.

Gli allineamenti visti finora però sono solo allineamenti orizzontali. Per ottenere un allineamento verticale, l'operatore che determina l'allineamento va prefissato con un punto (`.`). Il contenuto può essere allineato verticalmente in alto (`.<`), centrato verticalmente (`.^`), e allineato verticalmente in basso (`.>`). L'allineamento orizzontale e

verticale degli elementi può essere combinato, come mostra la tabella seguente:

| Allineamento | Orizzontale | | | |
|---|---|---|---|---|
| Verticale | - | Sinistra | Centro | Destra |
| | Alto | <.< | ^.< | >.< |
| | Centro | <.^ | ^.^ | >.^ |
| | Basso | <.> | ^.> | >.> |

Lo stile della colonna definisce se e quale trasformazione viene applicata alla colonna, e va messo come ultimo operatore per la colonna.