Markdown & Pandoc Essentials

Contents

简介

什么是 Markdown?

Markdown 是一种轻量级标记语言，你可以使用它向纯文本文档添加格式元素。Markdown 由 John Gruber 于 2004 年创建，现在是世界上最流行的标记语言之一。

使用 Markdown 与使用 WYSIWYG 编辑器不同。在 Microsoft Word 等应用程序中，你可以单击按钮来设置单词和短语的格式，并且更改会立即显示。Markdown 并非如此。当你创建一个 Markdown 格式的文件时，你向文本添加 Markdown 语法来指示哪些单词和短语应显示为不同格式。

例如，要表示标题，你可以在标题前添加一个井号（例如，# 标题一）。或者要使短语变为粗体，你可以在短语前后添加两个星号（例如，**此文本为粗体**）。在文本中看到 Markdown 语法可能需要一段时间才能适应，特别是如果你习惯于 WYSIWYG 应用程序。

你可以使用文本编辑器应用程序向纯文本文件添加 Markdown 格式元素。或者，你可以使用 macOS、Windows、Linux、iOS 和 Android 操作系统的众多 Markdown 应用程序之一。还有专门用于编写 Markdown 的几个基于 Web 的应用程序。

根据你使用的应用程序，你可能无法实时预览格式化的文档。但这没关系，Markdown 格式语法的主要设计目标是使其尽可能具有可读性。

为何使用 Markdown?

你可能想知道，人们为何使用 Markdown 而不是所见即所得编辑器。为什么在界面中按按钮格式化文本时还要使用 Markdown 来编写？事实证明，人们使用 Markdown 而不是所见即所得编辑器的原因有很多。

- Markdown 可用于一切。人们使用它来创建网站、文档、笔记、书籍、演示文稿、电子邮件和技术文档。
- Markdown 是可移植的。包含 Markdown 格式文本的文件几乎可以使用任何应用程序打开。如果你决定不喜欢当前使用的 Markdown 应用程序，你可以将 Markdown 文件导入另一个 Markdown 应用程序。这与 Microsoft Word 等将内容锁定为专有文件格式的文字处理应用程序形成了鲜明的对比。
- Markdown 与平台无关。你可以在运行任何操作系统的任何设备上创建 Markdown 格式的文本。
- Markdown 具有未来性。即使你使用的应用程序在未来某个时间点停止工作，你仍然可以使用文本编辑应用程序阅读 Markdown 格式的文本。对于需要无限期保存的书籍、大学论文和其他里程碑式文档，这是一个重要的考虑因素。
- Markdown 无处不在。像 Reddit 和 GitHub 这样的网站支持 Markdown，并且许多桌面和基于 Web 的应用程序也支持它。

Markdonw 是如何工作的?

当您使用 Markdown 书写时，文本会存储在具有.md 或.markdown 扩展名的纯文本文件中。但接下来呢？Markdown 格式的文件如何转换为 HTML 或可打印的文档？

简而言之，您需要一个能够处理 Markdown 文件的 Markdown 应用程序。

Markdown 应用程序使用称为 Markdown 处理器（通常也称为“解析器”或“实现”）的东西，将 Markdown 格式的文本提取出来并将其输出为 HTML 格式。在这一点上，您的文档可以在网络浏览器中查看，或与样式表结合使用并打印出来。您可以在下面看到此过程的可视化表示。注意

Markdown 应用程序和处理器是两个独立的组件。为了简洁起见，我在下图中将它们合并为一个元素（“Markdown 应用程序”）。

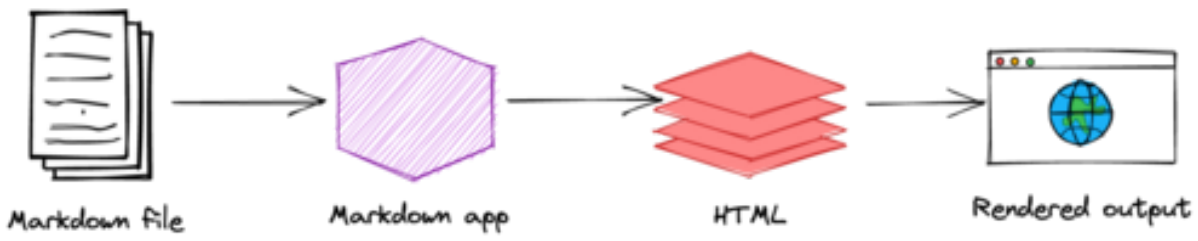

Figure 1: Markdown Flowchart

总而言之，这是一个由四部分组成的过程

1. 使用文本编辑器或专门的 Markdown 应用程序创建 Markdown 文件。该文件应具有.md 或.markdown 扩展名。
2. 在 Markdown 应用程序中打开 Markdown 文件。
3. 使用 Markdown 应用程序将 Markdown 文件转换为 HTML 文档。
4. 在网络浏览器中查看 HTML 文件，或使用 Markdown 应用程序将其转换为其他文件格式，例如 PDF。

Markdown 有什么用?

Markdown 是一种快速简便的方法，可用于做笔记、为网站创建内容以及生成可打印的文档。

学习 Markdown 语法并不需要很长时间，一旦您知道如何使用它，您就可以在几乎任何地方使用 Markdown 来书写。大多数人使用 Markdown 为网络创建内容，但 Markdown 适用于从电子邮件到购物清单的所有内容的格式化。

Markdown 的风格

使用 Markdown 最令人困惑的方面之一是几乎每个 Markdown 应用程序都实现了略有不同的 Markdown 版本。这些 Markdown 变体通常称为风格，如 GitHub Flavored Markdown，Pandoc Flavored Markdown 等等。

其他资源

有许多资源可用于学习 Markdown。以下是一些其他入门资源：

- **John Gruber 的 Markdown 文档** 由 Markdown 创建者编写的原始指南。
- **Markdown 教程** 一个开源网站，允许你在网络浏览器中尝试 Markdown。
- **Awesome Markdown** Markdown 工具和学习资源列表。
- **Markdown 排版** 一个多部分系列，描述了使用 pandoc 和 ConTeXt 排版 Markdown 文档的生态系统。

基本语法 Basic Syntax

概述

设计哲学

John Gruber 将 Markdown 设计为易于编写，并且更重要的是易于阅读：

> Markdown 格式的文档应该可以直接发布为纯文本，而不应看起来像是被标签或格式化指令标记过。

尽管 Markdown 的语法受到多种 text-to-HTML 过滤器的影响——包括 Setext、atx、Textile、reStructured-Text、Grutatext和EtText——但其语法灵感主要来源于纯文本电子邮件的格式。

因此，Markdown 的语法完全由标点符号组成，这些符号都经过精挑细选，使其外观能直观反映功能。例如：用星号包裹的文字看起来就像 *强调内容*；Markdown 列表看起来就是列表的形态；甚至区块引用也如同电子邮件的引文格式。

行内 HTML

Markdown 语法有且只有一个目的：成为适用于网络写作的格式。

Markdown 并非要替代 HTML，甚至不能与 HTML 相提并论。它的语法集非常精简，仅对应 HTML 标签的一小部分。其核心理念并非创造更便捷的 HTML 标签插入方式（我认为 HTML 标签本身已足够易用），而是让文本内容的阅读、书写和编辑变得更轻松。HTML 是发布格式，Markdown 是写作格式。因此 Markdown 的格式化语法仅处理纯文本能表达的内容。

对于 Markdown 语法未覆盖的标记，直接使用 HTML 标签[1]即可。无需特别声明或分隔标记来切换语法——直接使用标签就行。

唯一限制在于：块级 HTML 元素（如 、<table>、<pre>、<p>等）必须用空行与相邻内容分隔，且标签首尾不应有缩进。Markdown 会智能避免在块级 HTML 标签外围添加多余的<p>标签。

例如在 Markdown 文章中插入 HTML 表格：

```
这是普通段落。

<table>
    <tr>
        <td>Foo</td>
    </tr>
</table>

这是另一个普通段落。
```

需注意：Markdown 语法在块级 HTML 标签内部不会生效。例如不能在 HTML 区块内使用 Markdown 的*强调*语法。

而行级 HTML 标签（如、<cite>、<del>）可用于 Markdown 段落、列表项或标题中的任意位置。你甚至可以完全用 HTML 标签代替 Markdown 格式，比如用 HTML 的<a>或<img>标签替代 Markdown 的链接或图片语法。

与块级标签不同，Markdown 语法在行级 HTML 标签内部仍然有效。

特殊字符自动转义

HTML 中有两个字符需要特殊处理：< 和 &。左尖括号用于标签起始，& 符号用于表示 HTML 实体。若需将它们作为普通字符使用，必须转义为实体形式，如 < 和 &。

& 符号尤其令网络写作者头疼。若要在文中书写“AT&T”，必须写成“AT&T”。甚至在 URL 中也需要转义 & 符号。因此如果要链接到：

```
https://images.google.com/images?num=30&q=larry+bird
```

需将 URL 编码为：

```
https://images.google.com/images?num=30&amp;q=larry+bird
```

[1]Pandoc 在解析 Inline HTML 时是原封不动抄录，这在转化成 HTML 时是可以的，但在转化成 LaTeX 和 PDF 时是无法达到效果的，见：https://github.com/jgm/pandoc/issues/10043。

放在锚标签的 href 属性中。这极易被遗忘，堪称规范 HTML 文档中最常见的校验错误来源。

Markdown 允许你直接使用这些字符，并自动完成转义。当 & 符号作为 HTML 实体的一部分时将保留原状，否则会被转换为 &。

例如要在文章中插入版权符号：

```
&copy;
```

Markdown 将保持其不变。但若写作：

```
AT&T
```

Markdown 会将其转为：

```
AT&amp;T
```

同理，由于 Markdown 支持行内 HTML，若将尖括号作为 HTML 标签界定符，Markdown 会正常解析。但若写作：

```
4 < 5
```

Markdown 会将其转为：

```
4 &lt; 5
```

但在代码区间和代码块内，尖括号和 & 符号总是会被自动编码。这使得用 Markdown 撰写 HTML 教程变得轻松（与原生 HTML 相比，后者在演示 HTML 语法时极为笨拙，因为每个示例中的 < 和 & 都需要手动转义）。

块级元素和内联元素

在 HTML 中，元素主要分为两类：块级元素（Block Elements）和内联元素（Span Elements，也称为行内元素）。这两类元素在页面布局和行为上有显著区别。

Markdown 基本语法包括：

- 块级元素（Block elements）：paragraphs、line breaks、headers、blockquotes、lists、code blocks、horizontal rules；
- 内联元素（Span elements）：links、emphasis、code、images；
- 其它元素（Miscellaneous）：backslash escape、automatic links。

标题 Headings

Markdown 支持两种标题样式：Setext 和 atx。

Setext 风格的标题使用等号（一级标题）和短横线（二级标题）作为“下划线”。例如：

```
这是一级标题
==============

这是二级标题
--------------
```

任意数量的 = 或 - 都可以使用。

Atx 风格的标题在行首使用 1-6 个井号，对应 1-6 级标题。例如：

```
# Heading level 1
## Heading level 2
### Heading level 3
#### Heading level 4
##### Heading level 5
###### Heading level 6
```

你也可以使用“闭合的”atx 风格的标题。这纯粹是装饰性的——如果你觉得这样看起来更好看的话可以使用。闭合的井号数量甚至不需要与开头的井号数量匹配（标题级别由开头的井号数量决定）：

```
# 这是一级标题 #

## 这是二级标题 ##

### 这是三级标题 ######
```

段落 Paragraphs

段落就是由一个或多个连续的文本行组成，并由一个或多个空行分隔。（空行是指任何看起来像空行的行——仅包含空格或制表符的行也被视为空行。）

除非段落在列表中，否则不要用空格（spaces）或制表符（tabs）缩进段落。

```
It takes a great deal of bravery to stand up to our enemies,
but just as much to stand up to our friends.

It does not do to dwell on dreams and forget to live.

<!-- J.K. Rowling, Harry Potter and the Sorcerer's Stone -->
```

渲染效果如下：

It takes a great deal of bravery to stand up to our enemies, but just as much to stand up to our friends.

It does not do to dwell on dreams and forget to live.

换行 Line Breaks

在一行的末尾添加两个或多个空格，然后按回车键（return），即可创建一个换行（line break）或新行（
）。

如果你所使用的 Markdown 应用程序支持 HTML 的话[2]，你可以使用 HTML 的
 标签来实现换行。

```
You've gotta dance like there's nobody watching,
Love like you'll never be hurt,
Sing like there's nobody listening,
And live like it's heaven on earth.
<!-- William W. Purkey -->
```

渲染效果如下：

You've gotta dance like there's nobody watching,
Love like you'll never be hurt,
Sing like there's nobody listening,
And live like it's heaven on earth.

强调 Emphasis

通过将文本设置为粗体或斜体来强调其重要性。

粗体 Bold

要加粗文本，请在单词或短语的前后各添加两个星号（asterisks）或下划线（underscores）。如需加粗一个单词或短语的中间部分用以表示强调的话，请在要加粗部分的两侧各添加两个星号（asterisks）。

```
I just love **bold text**.

I just love __bold text__.

目前，**中国**为世界第二大经济体。

Love**is**bold
```

渲染结果如下：

I just love **bold text**.

I just love **bold text**.

目前，**中国**为世界第二大经济体。

[2]Pandoc 在转换成 latex 或 pdf 时不支持
 换行，见：https://github.com/jgm/pandoc/issues/10043。

Love**is**bold

注意

Markdown 应用程序在如何处理单词或短语中间的下划线上并不一致。为兼容考虑，在单词或短语中间部分加粗的话，请使用星号（asterisks）。

斜体 Italic

要用斜体显示文本，请在单词或短语前后添加一个星号（asterisk）或下划线（underscore）。要斜体突出单词的中间部分，请在字母前后各添加一个星号，中间不要带空格。

```
Italicized text is the *cat's meow*.

Italicized text is the _cat's meow_.

A*cat*meow
```

渲染结果如下：

Italicized text is the *cat's meow*.

Italicized text is the *cat's meow*.

A*cat*meow

注意

Markdown 的众多应用程序在如何处理单词中间的下划线上意见不一致。为了兼容起见，请用星号标注单词中间的斜体来表示着重。

粗斜体 Bold and Italic

要同时用粗体和斜体突出显示文本，请在单词或短语的前后各添加三个星号或下划线。要加粗并用斜体显示单词或短语的中间部分，请在要突出显示的部分前后各添加三个星号，中间不要带空格。

```
This text is ***really important***.

This text is ___really important___.

This text is __*really important*__.

This text is **_really important_**.

This is really***very***important text.
```

渲染结果如下：

This text is ***really important***.

This text is ***really important***.

This text is ***really important***.

This text is ***really important***.

This is really***very***important text.

注意

Markdown 应用程序在处理单词或短语中间添加的下划线上并不一致。为了实现兼容性，请使用星号将单词或短语的中间部分加粗并以斜体显示，以示重要。

引用块 Blockquotes

要创建块引用，请在段落前添加一个 > 符号。

```
> Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod
  tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam
  , quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo
  consequat.
```

渲染效果如下：

> Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.

多个段落的块引用

块引用可以包含多个段落。为段落之间的空白行各添加一个 > 符号。

```
> Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod
  tempor incididunt ut labore et dolore magna aliqua.
>
> Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut
  aliquip ex ea commodo consequat.
```

渲染效果如下：

> Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.
>
> Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.

嵌套块引用（Nested Blockquotes）

块引用可以嵌套。在要嵌套的段落前添加一个 >> (或者> >) 符号。

```
《三国演义》开篇写道：

> 词曰：
>
>> 滚滚长江东逝水，浪花淘尽英雄。是非成败转头空。青山依旧在，几度夕阳红。白发渔樵江渚上，惯看秋月春风。一壶浊酒喜相逢。古今多少事，都付笑谈中。
```

渲染效果如下：

《三国演义》开篇写道：

> 词曰：
>
> > 滚滚长江东逝水，浪花淘尽英雄。是非成败转头空。青山依旧在，几度夕阳红。白发渔樵江渚上，惯看秋月春风。一壶浊酒喜相逢。古今多少事，都付笑谈中。

带有其它元素的块引用（Blockquotes with Other Elements）

块引用可以包含其他 Markdown 格式的元素。并非所有元素都可以使用，你需要进行实验以查看哪些元素有效。

```
> *Everything* is going according to **plan**.
>
> - Revenue was off the chart.
> - Profits were higher than ever.
```

渲染效果如下：

> *Everything* is going according to **plan**.
>
> - Revenue was off the chart.
> - Profits were higher than ever.

注意

Pandoc 要求引用块必须前面必须要有空行，见扩展：`blank_before_blockquote`。

列表 Lists

Markdown 支持有序或无序列表 (ordered/numbered lists & unordered/bulleted lists)。

有序列表 Ordered Lists

要创建有序列表，请在每个列表项前添加数字并紧跟一个英文句点。数字**不必**按数学顺序排列，但是列表应当以数字 1 起始。

```
1. First item
2. Second item
3. Third item
4. Fourth item
    1. Indented item
    2. Indented item
5. Fourth item
```

渲染效果:

1. First item
2. Second item
3. Third item
4. Fourth item
    1. Indented item
    2. Indented item
5. Fourth item

```
If you instead wrote the list in Markdown like this:

1.  Bird
1.  McHale
1.  Parish

or even:

3. Bird
1. McHale
8. Parish
```

渲染效果:

If you instead wrote the list in Markdown like this:

1. Bird
2. McHale
3. Parish

or even:

3. Bird
4. McHale

5. Parish

CommonMark 和其它几种轻量级标记语言可以让你使用括号 ( )) 作为分隔符 (例如 1) `First item`)，但并非所有的 Markdown 应用程序都支持此种用法，因此，从兼容的角度来看，此用法不推荐。为了兼容起见，请只使用英文句点作为分隔符。

无序列表（Unordered Lists）

要创建无序列表，请在每个列表项前面添加破折号 (-)、星号 (*) 或加号 (+) 。缩进一个或多个列表项可创建嵌套列表。

```
- First item
- Second item
- Third item
- Fourth item

* First item
* Second item
* Third item
* Fourth item

+ First item
+ Second item
+ Third item
+ Fourth item

- First item
- Second item
- Third item
    - Indented item
    - Indented item
- Fourth item
```

渲染效果:

- First item
- Second item
- Third item
- Fourth item
- First item
- Second item
- Third item
- Fourth item
- First item

- Second item
- Third item
- Fourth item
- First item
- Second item
- Third item
    - Indented item
    - Indented item
- Fourth item

以数字开头的无序列表项 如果你需要以数字开头并且紧跟一个英文句号（也就是 .）的无序列表项，则可以使使用反斜线（\）来转义这个英文句号。

```
- 1968\. A great year!
- I think 1969 was second best.
```

渲染效果如下：

- 1968. A great year!
- I think 1969 was second best.

Markdown 应用程序在如何处理同一列表中混用不同分隔符上并不一致。为了兼容起见，请不要在同一个列表中混用不同的分隔符，最好选定一种分隔符并一直用下去。

在列表中添加元素

要在保留列表连续性的同时在列表中添加另一种元素，请将该元素缩进四个空格或一个制表符，如下例所示：

段落 列表中加入段落：

```
* This is the first list item.
* Here's the second list item.

    I need to add another paragraph below the second list item.

* And here's the third list item.
```

渲染效果如下：

- This is the first list item.
- Here's the second list item.

  I need to add another paragraph below the second list item.
- And here's the third list item.

引用 列表中加入引用：

```
* This is the first list item.
* Here's the second list item.

    > A blockquote would look great below the second list item.

* And here's the third list item.
```

渲染效果如下：

- This is the first list item.
- Here's the second list item.

  > A blockquote would look great below the second list item.
- And here's the third list item.

代码块 代码块通常缩进四个空格或一个制表符。当它们在列表中时，将它们缩进八个空格或两个制表符。

```
1. Open the file.
2. Find the following code block on line 21:

        <html>
          <head>
            <title>Test</title>
          </head>

3. Update the title to match the name of your website.
```

渲染效果如下：

1. Open the file.
2. Find the following code block on line 21:

   ```
   <html>
     <head>
       <title>Test</title>
     </head>
   ```
3. Update the title to match the name of your website.

图片 列表中加入图片：

```
1. Open the file containing the Linux mascot.
2. Marvel at its beauty.

    ![Tux, the Linux mascot](examples/basic-syntax/lists/../../../images/tux.
       png)

3. Close the file.
```

渲染效果如下：

1. Open the file containing the Linux mascot.
2. Marvel at its beauty.

Figure 2: Tux, the Linux mascot

3. Close the file.

列表 你可以在有序列表中嵌套无序列表，反之亦然。

```
1. First item
2. Second item
3. Third item
    - Indented item
    - Indented item
4. Fourth item
```

渲染效果如下：

1. First item
2. Second item
3. Third item
    - Indented item
    - Indented item
4. Fourth item

代码块 Code Blocks

代码块（Code blocks）通常采用四个空格或一个制表符缩进。当它们被放在列表中时，请将它们缩进八个空格或两个制表符。要创建不缩进行的代码块，请使用围栏代码块 (Fenced Code Blocks)。

```
The famous helloworld program for C language is :

    #include <stdio.h>

    int main(int argc, const char * argv[]) {
      printf("Hello, World!\n");
      return 0;
    }
```

渲染效果如下：

The famous helloworld program for C language is :

```
#include <stdio.h>

int main(int argc, const char * argv[]) {
  printf("Hello, World!\n");
  return 0;
}
```

转义反引号

如果你要表示为代码的单词或短语包含一个或多个反引号，你可以通过用双反引号 (“) 将单词或短语括起来来转义它。

```
``在你的 Markdown 文件中使用 `code`。``
```

渲染效果如下：

在你的Markdown 文件中使用 `code`。

代码块

要创建代码块，请将块的每一行至少缩进四个空格或一个制表符。

```
    <html>
      <head>
      </head>
    </html>
```

渲染效果如下：

```
<html>
  <head>
  </head>
</html>
```

分隔线 Horizontal Rules

要创建分隔线，请在单独一行上使用三个或多个星号 (***)、破折号 (---) 或下划线 (___) ，并且不能包含其他内容。

```
***

---

_________________
```

以上三个分隔线的渲染效果看起来都一样：

注意

为了兼容性，请在分隔线的前后均添加空白行。

链接 Links

要创建链接，请将链接文本括在方括号中，后面紧跟着括在圆括号中的 URL，如：

```
My favorite search engine is [Duck Duck Go](https://duckduckgo.com).
```

渲染效果如下：

My favorite search engine is Duck Duck Go.

添加标题

您可以选择为链接添加标题。当用户将鼠标悬停在链接上时，它将显示为工具提示。要添加标题，请在 URL 后用引号将其括起来，如：

```
My favorite search engine is [Duck Duck Go](https://duckduckgo.com "The best search engine for privacy").
```

渲染效果如下：

My favorite search engine is Duck Duck Go.

URL 和电子邮件地址

要快速将 URL 或电子邮件地址变成链接，请用尖括号将其括起来。

```
<https://www.markdownguide.org>

<fake@example.com>
```

渲染效果如下：

https://www.markdownguide.org

fake@example.com

格式化链接

要强调链接，请在方括号和圆括号前后添加星号。要将链接表示为代码，请在方括号中添加反引号。

```
I love supporting the **[EFF](https://eff.org)**.
This is the *[Markdown Guide](http://markdownguide.cn)*.
See the code on [**GitHub**](https://github.com/mattcone/markdown-guidecode).
```

渲染效果如下：

I love supporting the **EFF**. This is the *Markdown Guide*. See the code on **GitHub**.

引用式链接

假设您将一个 URL 作为标准 URL 链接添加到一个段落，并且在 Markdown 中看起来像这样：

```
In a hole in the ground there lived a hobbit. Not a nasty, dirty, wet hole,
    filled with the ends
of worms and an oozy smell, nor yet a dry, bare, sandy hole with nothing in it
    to sit down on or to
eat: it was a [hobbit-hole](https://en.wikipedia.org/wiki/Hobbit#Lifestyle "
    Hobbit lifestyles"), and that means comfort.
```

虽然它可能指向有趣的附加信息，但显示的 URL 除了使其更难阅读之外，实际上并没有给现有原始文本增加太多内容。要解决这个问题，您可以像这样格式化 URL：

```
In a hole in the ground there lived a hobbit. Not a nasty, dirty, wet hole,
    filled with the ends
of worms and an oozy smell, nor yet a dry, bare, sandy hole with nothing in it
    to sit down on or to
eat: it was a [hobbit-hole][h], and that means comfort.

[h]: <https://en.wikipedia.org/wiki/Hobbit#Lifestyle> "Hobbit lifestyles"
```

也可以采用隐式的链接名称：

```
In a hole in the ground there lived a hobbit. Not a nasty, dirty, wet hole,
    filled with the ends
of worms and an oozy smell, nor yet a dry, bare, sandy hole with nothing in it
    to sit down on or to
eat: it was a [hobbit-hole][], and that means comfort.

[hobbit-hole]: <https://en.wikipedia.org/wiki/Hobbit#Lifestyle> "Hobbit
    lifestyles"
```

在上面的实例中，呈现的输出将是相同的：

In a hole in the ground there lived a hobbit. Not a nasty, dirty, wet hole, filled with the ends of worms and an oozy smell, nor yet a dry, bare, sandy hole with nothing in it to sit down on or to eat: it was a hobbit-hole, and that means comfort.

注意

Markdown 应用程序对于如何处理 URL 中间的空格没有达成一致。为了兼容性，请尝试使用 %20 对 URL 中的任何空格进行 URL 编码。或者，如果您的 Markdown 应用程序支持 HTML，则可以使用 a HTML 标签。

URL 中间的括号也可能带来问题。为了兼容性，尝试使用 %28 对左括号 (() 进行 URL 编码，使用 %29 对右括号 ()) 进行 URL 编码。或者，如果 Markdown 应用程序支持 HTML，可以使用 a HTML 标签。

图片 Images

要添加图片的话，首先请添加感叹号 (!)，然后紧跟着是方括号，方括号中可添加备用文本 (alt text，即图片显示失败后显示此文本)，最后跟着圆括号，圆括号中添加图片资源的路径或 URL。你可以选择在

圆括号中的路径或 URL 之后添加标题（即 title 属性）。

```
1  ![The San Juan Mountains are beautiful!](chapters/basic-syntax/images/san-juan-
       mountains.jpg "San Juan Mountains")
```

渲染效果如下：

Figure 3: The San Juan Mountains are beautiful!

带链接的图片

要为图片添加链接，请先为图片的 Markdown 标记添加一个方括号，然后紧跟着一个圆括号，并在圆括号中添加链接地址。

```
1  [![An old rock in the desert](chapters/basic-syntax/../../images/shiprock.jpg "
       Shiprock, New Mexico by Beau Rogers")](https://www.flickr.com/photos/
       beaurogers/31833779864/in/photolist-Qv3rFw-34mt9F-a9Cmfy-5Ha3Zi-9msKdv-
       o3hgjr-hWpUte-4WMsJ1-KUQ8N-deshUb-vssBD-6CQci6-8AFCiD-zsJWT-nNfsgB-dPDwZJ-
       bn9JGn-5HtSXY-6CUhAL-a4UTXB-ugPum-KUPSo-fBLNm-6CUmpy-4WMsc9-8a7D3T-83KJev-6
       CQ2bK-nNusHJ-a78rQH-nw3NvT-7aq2qf-8wwBso-3nNceh-ugSKP-4mh4kh-bbeeqH-a7biME-
       q3PtTf-brFpgb-cg38zw-bXMZc-nJPELD-f58Lmo-bXMYG-bz8AAi-bxNtNT-bXMYi-bXMY6-
       bXMYv)
```

渲染效果如下：

转义字符 Escaping Characters

要显示原本用于格式化 Markdown 文档的字符，请在字符前面添加反斜杠字符 (\)。

```
\* 如果没有开头的反斜杠字符的话，这一行将显示为无序列表。
```

渲染效果如下：

* 如果没有开头的反斜杠字符的话，这一行将显示为无序列表。

可做转义的（英文）字符

以下列出的字符都可以通过使用反斜杠字符从而达到转义目的。

字符	名称
\	反斜杠（backslash）
‘	backtick (另请参见在代码中转义反引号)

字符	名称
*	星号（asterisk）
_	下划线（underscore）
{}	花括号（curly braces）
[]	方括号（brackets）
< >	angle brackets
()	圆括号或括号（parentheses）
#	井号（pound sign）
+	加号（plus sign）
-	减号（minus sign）(也叫连字符 hyphen)
.	句点（dot）
!	感叹号（exclamation mark）
|	管道符（pipe）(另请参见在表格中转义管道符)

HTML 标签

许多 Markdown 应用程序允许你在 Markdown 格式文本中使用 HTML 标签。如果你更喜欢某些 HTML 标签而不是 Markdown 语法，这会很有帮助。例如，有些人觉得使用 HTML 标签来处理图片更简单。当你需要更改元素的属性（例如指定文本颜色或更改图片的宽度）时，使用 HTML 也会很有帮助。

要使用 HTML，请将标签放置在 Markdown 格式文件文本中。

```
This **word** is bold. This <em>word</em> is italic.
```

渲染效果如下：

This **word** is bold. This word is italic.

注意

出于安全原因，并非所有 Markdown 应用程序都支持 Markdown 文档中的 HTML。如有疑问，请查看 Markdown 应用程序的文档。一些应用程序仅支持 HTML 标签的子集。

使用空行来分隔块级 HTML 元素，如 、<table>、<pre> 和 ，使其与周围内容分开。尽量不要使用制表符或空格缩进标签，因为这会影响格式。

您不能在块级 HTML 标签中使用 Markdown 语法。例如， italic and **bold** 将不起作用。

Pandoc's Markdown

概述 Overview

Pandoc 中支持扩展修订版本的 Markdown 语法。Markdown 的设计哲学指导了 Pandoc 在寻找表格、脚注和其他扩展的语法时所做的决定。然而，Pandoc 的目标与 Markdown 最初的目标不同。虽然 Markdown 最初是为生成 HTML 而设计的，但 Pandoc 是为多种输出格式而设计的。因此，虽然 Pandoc 允许嵌入原始 HTML，但它并不鼓励这样做，并提供了其他非 HTML 方式来表示定义列表、表格、数学和脚注等重要文档元素。

- 使用 Pandoc 中支持的 Markdown 语法用 `-f markdown`
- 使用标准 Markdown 语法用 `-f markdown_strict`

Pandoc 所支持的语法各种对标准 Markdown 语法的扩展可以通过在格式后以 `+EXTENSION` 添加或`-EXTENSION` 去除，如：

- `-f markdown-footnotes` 表示识别除了 `footnotes` 扩展之外的所有 pandoc Markdown 语法
- `-f markdown_strict+footnotes+pipe_tables` 表示识别标准 Markdown 语法加上 `footnotes` 和 `pipe_tables` 扩展语法。

段落

段落由一行或多行文本组成，后跟一行或多行空行。换行符被视为空格，因此您可以根据需要重新排列段落。如果需要强制换行，请在行尾添加两个或更多空格。

扩展：escaped_line_breaks

反斜杠后跟换行符也是一种硬换行符。注意：在多行和网格表格单元格中，这是创建硬换行符的唯一方法，因为单元格中的尾随空格会被忽略。

```
反斜杠后跟换行符 (`\`)\
也是一种硬换行符

+---------------+---------------+--------------------+
| Fruit         | Price         | Advantages         |
+===============+===============+====================+
| Bananas       | first line\   | first line\        |
|               | next line     | next line          |
+---------------+---------------+--------------------+
| Bananas       | first line\   | first line\        |
|               | next line     | next line          |
+---------------+---------------+--------------------+
```

渲染效果如下：

反斜杠后跟换行符 (\)
也是一种硬换行符

Fruit	Price	Advantages
Bananas	first line next line	first line next line
Bananas	first line next line	first line next line

标题

Pandoc 可以使用两种标题格式：Setext 和 ATX。

Setext 样式

Setext 样式的标题是一行带有“下划线”的文本，其中带有一行 = 符号（对于一级标题）或 - 符号（对于二级标题）。标题文本可能包括内联格式，例如强调和斜体。

```
A level-one heading
===================

A level-two heading
-------------------

A *formatted* heading
=====================

Introduction to [Makedown Guide](https://www.markdownguide.com)
---------------------------
```

ATX 格式

ATX 样式标题由 1 到 6 个连续的#符号和一行文本组成，在行尾可能有任意数量的符号。行首的符号#数量即为标题的级别。与 Setext 标题一样，ATX 标题允许内联格式。

```
# A level-one heading

## A level-two heading

### A *formatted* heading
```

```
6
7  #### Introduction to [Makedown Guide](https://www.markdownguide.com)
```

扩展：blank_before_header

原始 Markdown 语法不需要标题前有空行。Pandoc 确实需要这个（当然，文档的开头除外）。提出这一要求的原因是，# 很容易意外地出现在一行的开头（可能由于换行）。

```
1  I like several of their flavors of ice cream:
2  # 22, for example, and # 5.
```

扩展：space_in_atx_header

许多 Markdown 实现并不要求 ATX 标题开头的 # 与标题文本之间有空格，因此 # heading 1 和 #heading 1 都算作标题。Pandoc 默认要求# 与标题文本之间有空格，可以用 -f markdown-space_in_atx_header 来取消这个要求。

```
1  #head
2
3  # head
```

编译时运行：pandoc -f markdown-space_in_atx_header input.md -o output.md。

扩展：auto_identifiers

没有明确指定标识符的标题将根据标题文本自动分配唯一标识符。

从标题文本中获取标识符的默认算法是：

- 删除所有格式、链接等。
- 删除所有脚注。
- 删除所有非字母数字字符，下划线、连字符和句点除外。
- 用连字符替换所有空格和换行符。
- 将所有字母字符转换为小写。
- 删除第一个字母之前的所有内容（标识符不能以数字或标点符号开头）。
- 如果此后没有剩余内容，则使用标识符 section。

例如：

标题	标识符
Heading identifiers in HTML	heading-identifiers-in-html
Maître d'hôtel	maître-dhôtel
Dogs?–in *my* house?	dogs–in-my-house
[HTML], [S5], or [RTF]?	html-s5-or-rtf
3. Applications	applications
33	section

在大多数情况下，这些规则应该允许根据标题文本确定标识符。例外情况是多个标题具有相同的文本；在这种情况下，第一个标题将获得如上所述的标识符；第二个标题将获得相同的标识符并-1 附加；第三个标题将附加 -2；依此类推。

`gfm_auto_identifiers`（但是，如果启用，则会使用不同的算法；请参见下文。）

这些标识符用于在选项生成的目录中提供链接目标 `--toc`|`--table-of-contents`。它们还可以轻松地提供从文档某个部分到另一个部分的链接。例如，指向此部分的链接可能如下所示：

```
See the section on
[heading identifiers](#heading-identifiers-in-html-latex-and-context).
```

但请注意，这种提供章节链接的方法仅适用于 HTML、LaTeX 和 ConTeXt 格式。

如果 `--section-divs` 指定了该选项，则每个部分将被包裹在 section（或 div，如果 html4 指定了）中，并且标识符将附加到封闭的 <section>（或 ）标签而不是标题本身。这允许使用 JavaScript 操作整个部分，或在 CSS 中进行不同的处理。

扩展：ascii_identifiers

使生成的标识符为 `auto_identifiers` 纯 ASCII 码。带重音符号的拉丁字母中的重音符号会被去除，非拉丁字母会被省略。

扩展：gfm_auto_identifiers

更改 `auto_identifiers` 使用的算法以符合 GitHub 的方法。空格将转换为短划线 (-)，大写字母将转换为小写字母，除-和 _ 之外的标点符号将被删除。表情符号将替换为其名称。

扩展：header_attributes

可以在包含标题文本的行末尾使用以下语法为标题分配属性：

```
{#identifier .class .class key=value key=value}
```

请注意，虽然此语法允许分配类和键/值属性，但编写者通常不会使用所有这些信息。标识符、类和键/值属性用于 HTML 和基于 HTML 的格式（例如 EPUB 和 slidy）。标识符用于 LaTeX、ConTeXt、Textile、Jira 标记和 AsciiDoc 编写器中的标签和链接锚点。

即使指定了，带有类的标题 unnumbered 也不会被编号–number-sections。属性上下文中的单个连字符 (-) 相当于.unnumbered，并且在非英语文档中更可取。因此，

```
# My heading {-}
```

和

```
# My heading {.unnumbered}
```

如果存在 unnumbered 和 unlisted ，则标题将不会包含在目录中。（目前此功能仅适用于某些格式：基于 LaTeX 和 HTML、PowerPoint 和 RTF 的格式。）

```
# Heading identifiers in HTML {#foo}

你可以这样使用：

[标题 1](#foo)


# 中文标题 identifiers in HTML {#bar}

你可以这样使用：

[标题 2](#bar)
```

扩展：implicit_header_references

Pandoc 的行为就像每个标题都定义了引用链接（reference links）一样。因此，要链接到标题，只需要直接在标题加上[]。

如要链接到标题：

```
# Heading identifiers in HTML
```

你可以简单地写：

```
[Heading identifiers in HTML]
```

或者

[Heading identifiers in HTML][]

或者

[the section on heading identifiers][heading identifiers in HTML]

而不是明确给出标识符:

[Heading identifiers in HTML](#heading-identifiers-in-html)

如果有多个标题具有相同的文本，则相应的参考将仅链接到第一个标题，并且您需要使用明确的链接来链接到其他标题，如上所述。

与常规参考链接一样，这些参考不区分大小写。

显式链接引用定义始终优先于隐式标题引用。因此，在下面的例子中，链接将指向 bar，而不是 #foo：

```
# Foo

[foo]: bar

See [foo]
```

下面是一个完整示例：

```
# Heading identifiers in HTML

你可以简单地写

[Heading identifiers in HTML]

或者

[Heading identifiers in HTML][]

或者

[the section on heading identifiers][heading identifiers in
HTML]

而不是明确给出标识符:

[Heading identifiers in HTML](#heading-identifiers-in-html)


# 中文标题 identifiers in HTML

你可以简单地写

[中文标题 identifiers in HTML]

或者

[中文标题 identifiers in HTML][]

```

```
或者

[the section on heading identifiers][中文标题 identifiers in
HTML]

而不是明确给出标识符:

[中文标题 identifiers in HTML](#中文标题-identifiers-in-html)
```

引用块 Block quotations

Markdown 使用电子邮件约定来引用文本块。块引用是指一个或多个段落或其他块元素（例如列表或标题），每行前面都有一个 > 字符和一个可选的空格。（引用 > 不必从左边距开始，但缩进不应超过三个空格。）

```
> This is a block quote. This
> paragraph has two lines.
>
> 1. This is a list inside a block quote.
> 2. Second item.
```

渲染效果如下：

> This is a block quote. This paragraph has two lines.
>
> 1. This is a list inside a block quote.
> 2. Second item.

还允许使用“懒惰”形式，即 > 仅要求在每个块的第一行出现字符：

```
> This is a block quote. This
paragraph has two lines.

> 1. This is a list inside a block quote.
2. Second item.
```

渲染效果如下：

> This is a block quote. This paragraph has two lines.

> 1. This is a list inside a block quote.
> 2. Second item.

块引用中可以包含其他块引用的块元素。也就是说，块引用可以嵌套：

```
> This is a block quote.
>
> > A block quote within a block quote.
```

渲染效果如下：

> This is a block quote.
>
> > A block quote within a block quote.

如果该 > 字符后跟一个可选空格，则该空格将被视为块引用标记的一部分，而不是内容缩进的一部分。因此，要将缩进的代码块放入块引用中，你需要在 > 后添加五个空格：

```
含有代码的引用：

>     print('hello,world')
```

渲染效果如下：

含有代码的引用：

> ```
> print('hello,world')
> ```

扩展：blank_before_blockquote

原始 Markdown 语法并不要求块引用前有空行。Pandoc 则要求这样做（当然，文档开头除外）。这样做的原因是，a 很容易意外地 > 出现在行首（例如由于换行）。因此，除非 markdown_strict 使用这种格式，否则以下代码在 Pandoc 中不会生成嵌套的块引用：

```
block qoutes

> This is a block quote.
>
> > A block quote within a block quote.
```

渲染效果如下：

block qoutes

> This is a block quote.
>
> > A block quote within a block quote.

注意：上面的嵌套块引用中需要加入空行。不加入空行则不会变成块引用：

```
This is NOT a block quote.
Lorem Ipsum 始于西塞罗 (Cicero) 在公元前45年作的 “de Finibus Bonorum et
    Malorum” （善恶之尽）里 1.10.32 和 1.10.33章节。这本书是一本关于道德理论的
    论述，曾在文艺复兴时期非常流行。Lorem Ipsum 的第一行是：
> Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod
    tempor incididunt ut labore et dolore magna aliqua.
```

```
> This is a block quote.
>> Not nested, since `blank_before_blockquote` is enabled by default
```

代码块 Verbatim (code) blocks

缩进的代码块 Indented code blocks

缩进四个空格（或一个制表符）的文本块将被视为逐字文本：也就是说，特殊字符不会触发特殊格式，所有空格和换行符都会保留。例如：

```
示例代码：<!-- 代码块必须用空行与周围文本分隔开 -->

    if (a > 3) {
      moveShip(5 * gravity, DOWN);
    }
```

渲染效果如下：

示例代码：

```
if (a > 3) {
  moveShip(5 * gravity, DOWN);
}
```

初始（四个空格或一个制表符）缩进不被视为逐字文本的一部分，并将在输出中删除。

注意

逐字文本中的空白行不必以四个空格开头。

扩展：`fenced_code_blocks`

除了标准缩进的代码块外，Pandoc 还支持带围栏的代码块。这些代码块以一行三个或更多波浪号 (~) 开始，以一行波浪号结束，波浪号的长度必须至少与起始行一样长。这些行之间的所有内容都被视为代码。无需缩进：

```
if (a > 3) {
  moveShip(5 * gravity, DOWN);
}
```

与常规代码块一样，围栏代码块必须用空行与周围文本分隔开。

如果代码本身包含一行波浪号或反引号，则只需在开始和结束处使用一排较长的波浪号或反引号：

```
~~~~~~~~~~
code including tildes
~~~~~~~~~~
```

扩展：backtick_code_blocks

与 fenced_code_blocks 相同，但使用反引号 (`) 而不是波浪号 (~)。

扩展：fenced_code_attributes

或者，您可以使用以下语法将属性附加到隔离或反引号代码块：

```
```{#helloworld .c .numberLines startFrom="101" .lineAnchors}
#include <stdio.h>

int main(int argc, const char * argv[]) {
  printf("Hello, World!\n");
  return 0;
}
```
```

渲染效果如下：

```
101 #include <stdio.h>
102
103 int main(int argc, const char * argv[]) {
104   printf("Hello, World!\n");
105   return 0;
106 }
```

这里 #helloworld 是一个标识符，可以使用 [Helloworld](#helloworld) 建立链接；c 和 numberLines 是类，并且 startFrom 是一个值为 101 的属性。某些输出格式可以使用此信息进行语法高亮显示。目前，唯一使用此信息的输出格式是 HTML、LaTeX、Docx、Ms 和 PowerPoint。如果您的输出格式和语言支持高亮显示，则上面的代码块将高亮显示，并带有编号行。(要查看支持哪些语言，请输入 pandoc --list-highlight-languages。)

numberLines (或 number-lines) 类将使代码块的行以 1 或 startFrom 的属性值开始进行编号。lineAnchors (或 line-anchors) 类将使这些行在 HTML 输出中成为可点击的锚点。

要阻止所有高亮显示，请使用 --no-highlight 标志。要设置高亮显示样式，请使用 --highlight-style。

行块 Line Blocks

行块是由一系列以竖线 (|) 开头、后跟空格的行组成的序列。行的划分以及前导空格将在输出中保留；否则，这些行将被格式化为 Markdown 格式。这对于诗句和地址很有用：

```
| The limerick packs laughs anatomical
| In space that is quite economical.
|    But the good ones I've seen
```

```
|     So seldom are clean
| And the clean ones so seldom are comical

| 200 Main St.
| Berkeley, CA 94718
```

如果需要，可以将行硬换行，但续行必须以空格开头。

```
| The Right Honorable Most Venerable and Righteous Samuel L.
  Constable, Jr.
| 200 Main St.
| Berkeley, CA 94718
```

内容中允许使用行内格式（例如强调）（但不能跨越行边界）。块级格式（例如块引用或列表）无法识别。

此语法借用自 reStructuredText。

列表 Lists

无序列表

项目符号列表是由项目符号列表项组成的列表。项目符号列表项以项目符号（*、+ 或 -）开头，**列表前必须添加空行**。如：

```
 “紧凑”的列表：

<!-- 列表前必须添加空行 -->
* one
* two
* three

 “松散”的列表：

* one

* two

* three

 列表各行对齐看起来更美观：

* here is my first
  list item.
* and my second.

 但是 Markdown 允许“懒惰”的格式（不需要对齐）：

* here is my first
list item.
* and my second.
```

渲染效果：

“紧凑”的列表：

- one
- two
- three

“松散”的列表：

- one
- two
- three

列表各行对齐看起来更美观：

- here is my first list item.
- and my second.

但是 Markdown 允许“懒惰”的格式（不需要对齐）：

- here is my first list item.
- and my second.

列表项可以包含其他列表。在这种情况下，前面的空行是可选的。嵌套列表必须缩进，以与包含列表项的列表标记后的第一个非空格字符对齐。

列表项含有代码

```
* First paragraph.

  Continued.

* Second paragraph. With a code block, which must be indented with spaces:

      print('hello,world')
* `print('hello,world')`
```

渲染效果：

- First paragraph.

  Continued.

- Second paragraph. With a code block, which must be indented with spaces:

```
print('hello,world')
```

- `print('hello,world')`

嵌套列表

```
* fruits
  + apples
    - macintosh
    - red delicious
  + pears
  + peaches
* vegetables
  + broccoli
  + chard
```

渲染效果：

- fruits
  - apples
    - macintosh
    - red delicious
  - pears
  - peaches
- vegetables
  - broccoli
  - chard

有序列表

有序列表的工作方式与项目符号列表相同，只是项目以枚举器而不是项目符号开头。

在原始 Markdown 中，枚举器是十进制数字，后跟一个句点和一个空格。数字本身会被忽略，如：

```
有序列表 1：

1.  one
2.  two
3.  three

有序列表 2：

7.  one
```

```
5.  two
6.  three
```

渲染效果：

有序列表 1：

1. one
2. two
3. three

有序列表 2：

7. one
8. two
9. three

扩展：fancy_lists

与原始 Markdown 不同，Pandoc 允许使用大小写字母和罗马数字以及阿拉伯数字来标记有序列表项。列表标记可以用括号括起来，也可以后跟一个右括号或句点。它们必须与后面的文本至少间隔一个空格；如果列表标记是带句点的大写字母，则必须至少间隔两个空格。

该fancy_lists扩展还允许使用 # 代替数字作为有序列表标记：

```
fancy lists 1:

a. one
#. two

fancy lists 2:

b) one
#) two

fancy lists 3:

A.  one
#.  two

fancy lists 4:

#. one
#. two
```

渲染效果：

fancy lists 1:

a. one
b. two

fancy lists 2:

b) one
c) two

fancy lists 3:

A. one
B. two

fancy lists 4:

1. one
2. two

扩展：task_lists

Pandoc 支持任务列表，使用 GitHub-Flavored Markdown 的语法。

```
任务列表示例：

- [ ] an unchecked task list item
- [x] checked item
```

渲染效果：

任务列表示例：

- ☐ an unchecked task list item
- ☒ checked item

扩展：definition_lists

每个术语必须占一行，行尾可以空一行，并且必须跟一个或多个定义。定义以冒号或波浪号开头，可以缩进一至两个空格。

一个术语可能有多个定义，每个定义可以由一个或多个块元素（段落、代码块、列表等）组成，每个块元素缩进四个空格或一个制表位。定义的主体（不包括第一行）应该缩进四个空格。但是，与其他 Markdown 列表一样，除了段落或其他块元素的开头外，你可以“懒惰地”省略缩进。

```
Term 1

:   Definition 1

Term 2 with *inline markup*

:   Definition 2

        { some code, part of Definition 2 }

    Third paragraph of definition 2.
```

渲染效果：

Term 1 Definition 1

Term 2 with *inline markup* Definition 2

```
{ some code, part of Definition 2 }
```

Third paragraph of definition 2.

结束列表

如果您想在列表后放置缩进的代码块怎么办？要“截断”第二项之后的列表，您可以插入一些非缩进的内容，例如 HTML 注释，这样不会以任何格式产生可见的输出；如果您想要两个连续的列表而不是一个大列表，则可以使用相同的技巧：

```
-   item one
-   item two

<!-- end of list -->

    { my code block }


1.  one
2.  two
3.  three

<!-- end of list -->

1.  uno
5.  dos
6.  tres
```

渲染效果：

- item one
- item two

```
{ my code block }
```

1. one
2. two
3. three

1. uno
2. dos
3. tres

分隔线

Pandoc 的分隔线语法与基本语法是一致的。我们强烈建议水平线与周围文本之间使用空行分隔。如果水平线后没有空行，pandoc 可能会尝试将其后的行解释为 YAML 元数据块或表格。

表格

Pandoc 中支持四种列表样式，前三种要求等宽字体（fixed-width font）[3]，最后一种可以使用比例间隔字体（proportionally spaced font）[4]：

- `simple_tables`
- `multiline_tables`
- `grid_tables`
- `pipe_tables`

扩展：`table_captions`

所有 4 种表格类型均可选择添加标题（如下例所示）。标题是一个以字符串 `Table:`（或 `table:` 或仅 `:`）开头的段落，可以出现在表格之前或之后。

扩展：`simple_tables`

表头和表行必须各占一行。列对齐由表头文本相对于其下方虚线的位置决定[5]：

- 如果虚线与右侧的标题文本齐平，但左侧超出标题文本，则该列右对齐。

[3] Fixed-width Font（等宽字体），又称为 Monospaced Font，是一种所有字符（包括字母、数字、标点符号等）占据相同水平宽度的字体。
[4] Proportionally Spaced Font（比例间隔字体）是一种字符宽度不固定的字体，每个字符根据其实际形状占据不同的水平空间。
[5] 中文排版时该对齐是不好控制的，因为中文字体不属于等宽字体。

- 如果虚线与左侧的标题文本齐平，但在右侧超出标题文本，则该列左对齐。
- 如果虚线超出两侧的标题文本，则该列居中。
- 如果虚线与两侧的标题文本齐平，则使用默认对齐方式（在大多数情况下，将保持对齐）。

表格必须以一个空白行结束，或者以一行 dashes 后跟一个空白行结束。

可以省略列标题行，但表格结尾必须使用虚线。当省略标题行时，列对齐将根据表主体的第一行确定。

示例：

```
<!-- simple table -->

Table: 简单表格示例

  Right     Left     Center     Default
-------     ------ ----------   -------
     12     12        12            12
    123     123       123          123
      1     1          1             1

Table: 省略列标题行的简单列表示例

-------     ------ ----------   -------
     12     12        12            12
    123     123       123          123
      1     1          1             1
-------     ------ ----------   -------
```

渲染效果如下：

Table 4: 简单表格示例

Right	Left	Center	Default
12	12	12	12
123	123	123	123
1	1	1	1

Table 5: 省略列标题行的简单列表示例

12	12	12	12
123	123	123	123
1	1	1	1

扩展：mutiline_tables

多行表格允许标题行和表格行跨越多行文本（但不支持跨越表格多列或多行的单元格）。

多行表格的工作方式类似于简单表格，但有以下区别：

- 必须以一行 dashes 开头，位于标题文本之前（除非省略标题行）。
- 必须以一排 dashes 结尾，然后是一个空行。
- 行与行之间必须用空行分隔。

在多行表格中，表格解析器会关注列的宽度，编写器会尝试在输出中重现这些相对宽度。因此，如果您发现输出中某一列太窄，请尝试在 Markdown 源代码中将其加宽。

在多行表格可以省略表头。

多行表可能只有一行，但该行后面应该跟着一个空行（然后是结束表格的 dashes 行），否则该表可能会被解释为简单表格。

示例：

```
-------------------------------------------------------------
 Centered   Default           Right Left
  Header    Aligned         Aligned Aligned
----------- ------- --------------- -------------------------
   First    row                12.0 Example of a row that
                                    spans multiple lines.

  Second    row                 5.0 Here's another one. Note
                                    the blank line between
                                    rows.
-------------------------------------------------------------
Table: Here's the caption. It, too, may span
multiple lines.

----------- ------- --------------- -------------------------
   First    row                12.0 Example of a row that
                                    spans multiple lines.

  Second    row                 5.0 Here's another one. Note
                                    the blank line between
                                    rows.
----------- ------- --------------- -------------------------

: Here's a multiline table without a header.

----------- ------- --------------- -------------------------
   First    row                12.0 Example of a row that
                                    spans multiple lines.

----------- ------- --------------- -------------------------

: Here's a multiline table with just one row.
```

渲染效果如下：

Table 6: Here's the caption. It, too, may span multiple lines.

Centered Header	Default Aligned	Right Aligned	Left Aligned
First	row	12.0	Example of a row that spans multiple lines.
Second	row	5.0	Here's another one. Note the blank line between rows.

Table 7: Here's a multiline table without a header.

First	row	12.0	Example of a row that spans multiple lines.
Second	row	5.0	Here's another one. Note the blank line between rows.

Table 8: Here's a multiline table with just one row.

First	row	12.0	Example of a row that spans multiple lines.

扩展：`grid_tables`

= 行将表头与表体分隔开来，对于无表头的表格，可以省略该行。网格表格的单元格可以包含任意块元素（多个段落、代码块、列表等）。

单元格可以跨越多列或多行。

表头可能包含多行。

可以像管道表格一样指定对齐方式，通过在标题后的分隔线边界处放置冒号 `:`。

对于无表头的表，冒号位于顶行。

可以通过使用分隔线（=而不是-）来定义表格脚注。

表格脚注必须始终放在表格的最底部。

可以使用 Emacs 的表格模式 (M-x table-insert) 轻松创建网格表。

示例：

```
: 简单的网格表格

+---------------+---------------+--------------------+
| Fruit         | Price         | Advantages         |
+===============+===============+====================+
| Bananas       | $1.34         | - built-in wrapper |
|               |               | - bright color     |
+---------------+---------------+--------------------+
| Oranges       | $2.10         | - cures scurvy     |
|               |               | - tasty            |
+---------------+---------------+--------------------+

: 单元格可以跨越多列或多行

+---------------------+----------+
| Property            | Earth    |
+=============+=======+==========+
|             | min   | -89.2 °C |
| Temperature +-------+----------+
| 1961-1990   | mean  | 14 °C    |
|             +-------+----------+
|             | max   | 56.7 °C  |
+-------------+-------+----------+

: 表头可能包含多行

+----------------------+-----------------------+
| Location             | Temperature 1961-1990 |
|                      | in degree Celsius     |
|                      +-------+-------+-------+
|                      | min   | mean  | max   |
+======================+=======+=======+=======+
| Antarctica           | -89.2 | N/A   | 19.8  |
+----------------------+-------+-------+-------+
| Earth                | -89.2 | 14    | 56.7  |
+----------------------+-------+-------+-------+

: 单元格对齐方式

+---------------+---------------+--------------------+
| Right         | Left          | Centered           |
+==============:+:==============+:==================:+
| Bananas       | $1.34         | built-in wrapper   |
+---------------+---------------+--------------------+

: 无表头的表格对齐

+--------------:+:--------------+:------------------:+
| Right         | Left          | Centered           |
+---------------+---------------+--------------------+

```

```
: 表格脚注

+---------------+---------------+
| Fruit         | Price         |
+===============+===============+
| Bananas       | $1.34         |
+---------------+---------------+
| Oranges       | $2.10         |
+===============+===============+
| Sum           | $3.44         |
+===============+===============+
```

渲染效果如下：

Table 9: 简单的网格表格

Fruit	Price	Advantages
Bananas	$1.34	• built-in wrapper • bright color
Oranges	$2.10	• cures scurvy • tasty

Table 10: 单元格可以跨越多列或多行

Property		Earth
Temperature 1961-1990	min	-89.2 °C
	mean	14 °C
	max	56.7 °C

Table 11: 表头可能包含多行

Location	Temperature 1961-1990 in degree Celsius		
	min	mean	max
Antarctica	-89.2	N/A	19.8
Earth	-89.2	14	56.7

Table 12: 单元格对齐方式

Right	Left	Centered
Bananas	$1.34	built-in wrapper

Table 13: 无表头的表格对齐

Right	Left	Centered

Table 14: 表格脚注

Fruit	Price
Bananas	$1.34
Oranges	$2.10
Sum	$3.44

扩展：`pipe_tables`

管道表格语法与 PHP Markdown Extra 表格 表相同。起始和结束竖线字符（|）是可选的，但所有列之间都需要竖线。冒号指示列对齐方式。**表头不可省略**。要模拟无表头表格，请添加一个带有空白单元格的表头。由于管道指示列边界，因此列无需像上例那样垂直对齐。

管道表格的单元格不能包含段落和列表等块元素，也不能跨越多行。如果 Markdown 源代码中的任何一行比列宽（参见`--columns`），则表格将占据整个文本宽度，单元格内容将换行，相对单元格宽度由表格标题和表格主体之间分隔线的虚线数量决定。（例如，`---|-`将第一列设置为整个文本宽度的 3/4，第二列设置为整个文本宽度的 1/4。）另一方面，如果没有行比列宽更宽，则单元格内容将不会换行，单元格将根据其内容调整大小。

示例：

```
: Demonstration of pipe table syntax.

| Right | Left | Default | Center |
| ----: | :--- | ------- | :----: |
|    12 | 12   | 12      |   12   |
```

```
|   123 | 123  | 123      |  123   |
|     1 | 1    | 1        |   1    |

: 在 Markdown 代码中列垂直对齐的管道表格

| fruit  | price |
| ------ | ----: |
| apple  |  2.05 |
| pear   |  1.37 |
| orange |  3.09 |

: 在 Markdown 代码中没有列垂直对齐的管道表格

fruit| price
-----|-----:
apple|2.05
pear|1.37
orange|3.09

: 没有表头的管道表格

|
| ------ | ----: |
| apple  |  2.05 |
| pear   |  1.37 |
| orange |  3.09 |
```

渲染效果如下：

Table 15: Demonstration of pipe table syntax.

Right	Left	Default	Center
12	12	12	12
123	123	123	123
1	1	1	1

Table 16: 在 Markdown 代码中列垂直对齐的管道表格

fruit	price
apple	2.05
pear	1.37
orange	3.09

Table 17: 在 Markdown 代码中没有列垂直对齐的管道表格

fruit	price
apple	2.05
pear	1.37
orange	3.09

Table 18: 没有表头的管道表格

apple	2.05
pear	1.37
orange	3.09

内联格式 Inline formatting

强调

- 斜体：使用 * 或 _ 包裹
- 粗体：使用 ** 或 __ 包裹
- * 或 _ 被空格包围或用反斜杠转义不会触发强调

例如：

```
This text is _emphasized with underscores_, and this
is *emphasized with asterisks*.

This is **strong emphasis** and __with underscores__.

This is * not emphasized *, and \*neither is this\*.
```

渲染效果如下：

This text is *emphasized with underscores*, and this is *emphasized with asterisks*.

This is **strong emphasis** and **with underscores**.

This is * not emphasized *, and *neither is this*.

扩展：intraword_underscores

由于 _ 有时会在单词和标识符内部使用，因此 pandoc 不会将 _ 字母数字字符包围的解释为强调标记。如果只想强调单词的一部分，请使用 *：

```
feas*ible*, not feas*able*.
```

渲染效果如下：

feas*ible*, not feas*able*.

Pandoc 默认是启用该扩展的，如果要想将单词内部的 _ 解析强调语法，需要取消该扩展：`-f markdown-intraword_underscores`。

扩展：strikeout

要用水平线划掉一段文本，请以开始和结束该部分 ~~。例如，

```
This ~~is deleted text.~~
```

渲染效果如下：

This ~~is deleted text.~~

扩展：superscript 和 subscript

上标可以用字符包围上标文本来书写^；下标也可以用字符包围下标文本来书写~。例如，

```
H~2~O is a liquid.  2^10^ is 1024.
```

渲染效果如下：

H_2O is a liquid. 2^{10} is 1024.

^...^ 或之间的文本 ~...~ 不能包含空格或换行符。如果上标或下标文本包含空格，则必须使用反斜杠转义这些空格。（这是为了防止在日常使用 ~ 和时意外地将文本变为上标或下标 ^，以及与脚注产生不良交互。）因此，如果您希望下标中包含字母 P 和“a cat”，请使用 `P~a\ cat~`，而不是 `P~a cat~`。

抄录（代码）

- 要抄录一小段文本，请将其放在反引号内
- 如果抄录文本包含反引号，请使用双反引号
- 反斜杠转义符（和其他 Markdown 结构）在抄录环境中不起作用

```
What is the difference between `>>=` and `>>`?

Here is a literal backtick `` ` ``.

This is a backslash followed by an asterisk: `\*`.
```

渲染效果如下：

What is the difference between >>= and >>?

Here is a literal backtick `.

This is a backslash followed by an asterisk: *.

扩展：inline_code_attributes

可以将属性附加到逐字文本，就像围栏代码块一样：

```
<!-- pandoc --list-highlight-languages 查看支持的语言 -->
| 语言     | HelloWorld 语句                    |
| :--:    | :-----                            |
| C       | `printf("hello,world")`{.c}       |
| C++     | `cout << "hello,world"`{.cpp}     |
| Java    | `println("hello,world")`{.java}   |
| Python  | `print('hello,world')`{.python}   |
```

渲染效果如下：

语言	HelloWorld 语句
C	`printf("hello,world")`
C++	`cout << "hello,world"`
Java	`println("hello,world")`
Python	`print('hello,world')`

下划线

要为文本添加下划线，请使用以下 underline 类：

```
[Underline]{.underline}
```

渲染效果如下：

Underline

Small Caps

要编写小型大写字母，请使用以下 smallcaps 类：

```
[Small caps]{.smallcaps}
```

渲染效果如下：

SMALL CAPS

高亮显示

要突出显示文本，请使用以下 mark 类：

```
[Mark]{.mark}
```

渲染效果如下：

Mark

链接 Links

使用链接有下面几种形式：

- 自动链接：<URL>
- 内联链接：[链接文本](URL) 或 [链接文本](URL "链接标题")
- [链接文本][链接标签]
- 隐式链接（“链接文本”和”链接标题”相同）：[链接文本][]
- 快捷链接（“链接文本”和”链接标题”相同）：[链接文本]
- [链接文本](#标题标识符) 或 [标题][6]

“链接标签”的定义：

- [链接标签]: URL
- [链接标签]: URL "链接标题"
- [链接标签]: URL (链接标题)
- [链接标签]: <URL>

自动链接 Automatic Links

如果将 URL 或电子邮件地址放在尖括号中，它将变成链接：

[6]标题标识符可以是 Pandoc 按一定规则自动生成，也可以手动定义。

```
<https://daringfireball.net/projects/markdown/syntax>

<https://www.markdownguide.org/>
```

渲染结果如下：

https://daringfireball.net/projects/markdown/syntax

https://www.markdownguide.org/

内联链接 Inline Links

内联链接由方括号内的链接文本和括号内的 URL[7] 组成。(URL 后面也可以跟着链接标题，并用引号引起来。)

```
The [Markdown Guide](https://www.markdownguide.org/ "Click to Visit Markdown
    Guide") is a free and open-source reference guide that explains how to use
    Markdown, the simple and easy-to-use markup language you can use to format
    virtually any document.
```

渲染结果如下：

The Markdown Guide is a free and open-source reference guide that explains how to use Markdown, the simple and easy-to-use markup language you can use to format virtually any document.

方括号和圆括号之间不能有空格。链接文本可以包含格式（例如强调），但标题不能。

内联链接中的电子邮件地址无法自动检测，因此必须加上前缀 `mailto`：

```
[Write me!](mailto:sam@green.eggs.ham)
```

引用链接 Reference Links

显式引用链接包含两部分：链接本身和链接定义，后者可能出现在文档的其他位置（链接之前或之后）。

链接由方括号内的链接文本和方括号内的标签组成。(除非启用了扩展 `spaced_reference_links`，否则两者之间不能有空格。)

链接定义由方括号内的标签、冒号和空格、URL 以及可选的（空格后）链接标题（用引号或括号括起来）组成。

标签不能被解析为文献引用（假设启用了扩展 `citations`）：**文献引用优先于链接标签**。

URL 可以选择性地用尖括号括起来。

标题可以放在下一行。

[7] URL 可以是网址也可以是本地路径。

请注意，链接标签不区分大小写。

在隐式引用链接中，第二对括号为空。

以下是一些示例：

```
- 设计哲学：[Markdown Philosophy]
- 基本语法：[Markdown Syntax]
- 基本语法：[Basic Syntax][BASIC]<!-- 请注意，链接标签不区分大小写 -->
- 元素：[Block Elements]
- Markdown 指南：[Markdown Guide]
- 扩展语法：[Extended Syntax]
- 速查表：[Cheat Sheet][]<!-- 在隐式引用链接中，第二对括号为空 -->

[Markdown Philosophy]: https://daringfireball.net/projects/markdown/syntax#philosophy  "Philosophy of Markdown"
[Block Elements]: https://daringfireball.net/projects/markdown/syntax#block 'Block Elements'
[Markdown Syntax]: https://daringfireball.net/projects/markdown/syntax
[Markdown Guide]: https://www.markdownguide.org/ (Markdown Guide)
[Extended Syntax]: <https://www.markdownguide.org/extended-syntax/>
[basic]: https://www.markdownguide.org/basic-syntax/
[Cheat Sheet]: https://www.markdownguide.org/cheat-sheet/
```

渲染效果如下：

- 设计哲学：Markdown Philosophy
- 基本语法：Markdown Syntax
- 基本语法：Basic Syntax
- 元素：Block Elements
- Markdown 指南：Markdown Guide
- 扩展语法：Extended Syntax
- 速查表：Cheat Sheet

注意

在 Markdown.pl Markdown 和大多数其他实现中，引用链接定义不能出现在嵌套结构中，例如列表项或块引用。Pandoc 取消了这一限制。因此，以下代码在 Pandoc 中可以正常工作，但在大多数其他实现中则不行：

> The most visited search egnine is google.

The top 3 search engines:

- google
- yandex
- baidu

扩展：shortcut_reference_links

在快捷引用链接中，第二对括号可以完全省略：

```
See [my website].

[my website]: http://foo.bar.baz
```

内部链接 Internal Links

要链接到同一文档的其他部分，请使用自动生成的标识符（请参阅标题标识符）。例如：

```
See the [Introduction].
```

或者

```
See the [Introduction](#introduction).
```

或者

```
See the [Introduction].

[Introduction]: #introduction
```

内部链接目前支持 HTML 格式（包括 HTML 幻灯片和 EPUB）、LaTeX 和 ConTeXt。

图片

在链接语法前加上！就是图片插入语法[8]。“链接文本”将用作图片的替代文本，图片的标题是“链接文本”内容，“链接标题”在鼠标悬停在图片上时显示的文本（见扩展: implicit_figures）。如：

```
<!-- 使用图片：在链接前加上感叹号！ -->

<!-- 内联形式："标题" 在鼠标悬停在图上时显示 -->
![内联形式：翠鸟（Eisvogel）](examples/pandoc-flavored-markdown/../../images/
    eisvogel-bayern.jpg "翠鸟图片")

<!-- 引用形式 -->
![引用形式：翠鸟（Eisvogel）][img-ref]

[img-ref]: examples/pandoc-flavored-markdown/../../images/eisvogel-bayern.jpg "
    翠鸟图片"

<!-- 隐式形式 -->
![隐式形式：翠鸟（Eisvogel）][]

```

[8]Markdown-Include 的 Makefile 只能处理“内联形式”的图片地址，无法处理“引用形式”的图片地址。

```
14  [隐式形式: 翠鸟 (Eisvogel)]: examples/pandoc-flavored-markdown/../../images/
        eisvogel-bayern.jpg "翠鸟图片"
15
16  <!-- 快捷形式 -->
17  ![快捷形式: 翠鸟 (Eisvogel)]
18
19  [快捷形式: 翠鸟 (Eisvogel)]: examples/pandoc-flavored-markdown/../../images/
        eisvogel-bayern.jpg "翠鸟图片"
```

渲染效果如下:

Figure 4: 内联形式: 翠鸟 (Eisvogel)

Figure 5: 引用形式: 翠鸟（Eisvogel）

Figure 6: 隐式形式: 翠鸟（Eisvogel）

Figure 7: 快捷形式: 翠鸟（Eisvogel）

扩展: `implicit_figures`

如果图片带有非空的 alt 文本，且单独出现在段落中，则该图片将被渲染为带有标题的图形。图片的 alt 文本将用作标题。

```
1  ![This is the caption](/url/of/image.png)
```

如何渲染取决于输出格式。某些输出格式（例如 RTF）尚不支持图形。在这些格式中，您只会在段落中看到一张图片，没有标题。

如果你只是想要一个普通的内联图片，请确保它不是段落中唯一的内容。一种方法是在图片后插入一个不间断的空格:

```
1  ![This image won't be a figure](/url/of/image.png)\
```

请注意，在 reveal.js 幻灯片中，段落中具有该类的图像本身 r-stretch 将填满屏幕，并且标题和图形标签将被省略。

扩展：link_attributes

图片的 width 和 height 属性会被特殊处理。如果不带单位使用，则单位被假定为像素（pixels）。但是，可以使用以下任何单位标识符：px、cm、mm、in、inch 和 %。**数字和单位之间不得有任何空格**。例如：

```
1 ![](chapters/pandoc-flavored-markdown/file.jpg){ width=50% }
```

- 尺寸可能会转换为与输出格式兼容的形式（例如，将 HTML 转换为 LaTeX 时，以像素为单位的尺寸将转换为英寸）。像素与物理测量值之间的转换受 --dpi 选项影响（默认情况下，假定为 96 dpi，除非图像本身包含 dpi 信息）。

- 单位 % 通常与某个可用空间相关。例如，上面的示例将渲染成以下内容。

  - HTML：
  - LaTeX：\includegraphics[width=0.5\textwidth,height=\textheight]{file.jpg}（如果您使用自定义模板，则需要 graphicx 按照默认模板进行配置。）
  - ConTeXt：\externalfigure[file.jpg][width=0.5\textwidth]

- 一些输出格式具有类（ConTeXt）或唯一标识符（LaTeX \caption）或两者（HTML）。

- 当未指定 width 或 height 属性时，后备方法是查看图像分辨率和图像文件中嵌入的 dpi 元数据。

示例：

```
1 ![eisvogel](examples/pandoc-flavored-markdown/../../images/eisvogel-bayern.jpg)
     { width=50% }
2
3 ![eisvogel](examples/pandoc-flavored-markdown/../../images/eisvogel-bayern.jpg)
     { width=150px }
4
5 ![eisvogel](examples/pandoc-flavored-markdown/../../images/eisvogel-bayern.jpg)
     { height=160px }
```

渲染效果如下：

Figure 8: eisvogel

Figure 9: eisvogel

Figure 10: eisvogel

脚注

扩展：`footnotes`

Pandoc 的 Markdown 允许使用脚注，使用以下语法：

```
Here is a footnote reference,[^1] and another.[^longnote]

```

```
[^1]: Here is the footnote.

[^longnote]: Here's one with multiple blocks.

    Subsequent paragraphs are indented to show that they
belong to the previous footnote.

        { some.code }

    The whole paragraph can be indented, or just the first
    line.  In this way, multi-paragraph footnotes work like
    multi-paragraph list items.

This paragraph won't be part of the note, because it
isn't indented.
```

脚注引用中的标识符不得包含空格、制表符、换行符或字符 ^、[、或]。这些标识符仅用于将脚注引用与注释本身关联起来；在输出中，脚注将按顺序编号。

脚注本身不必放在文档末尾。它们可以出现在除其他块元素（列表、块引用、表格等）之外的任何位置。每个脚注都应与周围内容（包括其他脚注）用空行分隔。

扩展：inline_notes

行内脚注也是允许的（不过，与常规注释不同，行内脚注不能包含多个段落）。语法如下：

```
Here is an inline note.^[Inline notes are easier to write, since
you don't have to pick an identifier and move down to type the
note.]
```

行内脚注和常规脚注可以自由混合。

HTML 代码

扩展：raw_html

可以直接在文档中插入 HTML 代码（除了代码块等，其中 <,> 和 & 符不会被翻译）。因为标准 Markdown 允许插入 HTML，但是 Pandoc 为了可以根据需要禁用 HTML，将其设计成一个扩展。

扩展：markdown_in_html_blocks

使用 `markdown_strict` 格式的时候，HTML 代码中的 Markdown 语法不会被翻译，但是使用 Pandoc 的 Markdown 格式 markdown 格式时，HTML 代码中的 Markdown 语法也会被翻译，但是有一个例外，HTML 代码 <script> 和 <style> 标签中的 Markdown 语法也不会被翻译。

例如，Pandoc 会将

```
<table>
<tr>
<td>*one*</td>
<td>[a link](https://google.com)</td>
</tr>
</table>
```

转换成：

```
<table>
<tr>
<td><em>one</em></td>
<td><a href="https://google.com">a link</a></td>
</tr>
</table>
```

LaTeX/TeX 代码

扩展：`raw_tex`

除了原始 HTML 之外，Pandoc 还允许将原始 LaTeX、TeX 和 ConTeXt 代码包含在文档中。内联 TeX 命令将被保留，并原封不动地传递给 LaTeX 和 ConTeXt 编写器。因此，例如，您可以使用 LaTeX 来包含 BibTeX 引用：

```
This result was proved in \cite{jones.1967}.
```

请注意，在 LaTeX 环境中，例如

```
\begin{tabular}{|l|l|}\hline
Age & Frequency \\ \hline
18--25  & 15 \\
26--35  & 33 \\
36--45  & 22 \\ \hline
\end{tabular}
```

开始和结束标签之间的材料将被解释为原始 LaTeX，而不是 Markdown。

使用元数据 Metadata

YAML 介绍

YAML 最初是“Yet Another Markup Language”的缩写，但后来为了强调其数据导向（而非标记语言）的特性，官方将其重新定义为“YAML Ain’t Markup Language”（一种递归缩写，即“YAML 不是标记语言”）。最初设计时，YAML 被视作类似 XML 的标记语言，但后来转向更简洁的数据描述格式，专注于存储和传输

结构化数据（如配置、API 请求等）。与 XML（标记语言）不同，YAML 避免使用标签（<>），而是依赖缩进和符号表示结构。

在 Pandoc 中，YAML 元数据是指文档开头的一段 YAML 格式的内容，用于描述文档的属性和元信息。通过读取 Pandoc 标记文件中的 YAML 元数据，我们可以获取文档的各种属性，如标题、作者、日期等，以及自定义的元信息。

当 Pandoc 生成独立文档时，会利用标题、作者等元数据来填充部分信息。这些数据通常不会在 Markdown 源文件中直接指定——毕竟 Markdown 语法本身并未提供定义此类元数据的标注方式——但您可以通过 `--metadata` 选项或 YAML 格式进行设置（具体方法见下文说明）。

严格来说，生成输出时会用到两种类型的变量：

- 通过 `--metadata` 指定的元数据 (metadata)，
- 通过 `--variable` 指定的模板变量 (variable)。

二者的核心区别在于：元数据可被 Pandoc 及其过滤器（即在最终格式化前处理输入内容的脚本）识别和处理，而模板变量仅作用于模板系统。若您通过 `--metadata` 标签或元数据头文件设置某个变量，该变量同样可供模板系统调用，因此通常只需使用元数据即可。虽然最终输出可能存在细微差异（因为过滤器可能对元数据进行特殊处理而忽略普通变量），但坚持使用单一类型选项显然更简便。除非有充分理由，否则建议统一采用元数据设置方式。

Pandoc 读取 YAML 元数据

读取 Pandoc 标记文件中的 YAML 元数据可以通过使用 Pandoc 提供的命令行参数或 API 来实现。以下是一个示例命令行使用方式：

```
pandoc --metadata-file=metadata.yaml input.md -o output.html
```

上述命令中，metadata.yaml 是包含 YAML 元数据的文件，input.md 是待转换的标记文件，output.html 是转换后的输出文件。

在实际应用中，读取 Pandoc 标记文件中的 YAML 元数据可以用于自动化文档处理、生成静态网页、构建电子书等场景。通过提取元数据，我们可以根据文档属性进行个性化处理，如根据作者生成不同样式的页面，根据日期进行归档等。

Pandoc 命令中的几个相关选项：

`--metadata-file=FILE` Read metadata from the supplied YAML (or JSON) file.

`-M KEY[=VAL], --metadata=KEY[:VAL]` Set the metadata field KEY to the value VAL.

`-V KEY[=VAL], --variable=KEY[:VAL]` Set the template variable KEY to the string value VAL when rendering the document in standalone mode.

注意：将“CJKmainfont: 方正楷体_GBK”写入 metadata.yaml 时，会被解析成“方正楷体_GBK”，导致 LaTeX 编译出错。这是因为 Pandoc 将其解析为 Markdown 格式[9]，可以将相关代码直接以 `header-includes` 形式加入。

数学公式 Mathematical Expressions

数学公式包含两种：

Inline math 即**行内公式**或**内联公式**，指的是将数学公式嵌入到文本段落的行内，与其他文本内容混合排列。在 LaTeX 中，行内公式通常用单个美元符号 `$` 包裹。

Displayed math 即**行间公式**或**显示公式**，指的是将数学公式单独占据一行显示，通常会居中排列，并且比行内公式具有更大的视觉突出性。在 LaTeX 中，行间公式通常用双美元符号 `$$` 包裹。

总而言之，行内公式用于在文本中简短地插入数学表达式，而行间公式则用于更重要或更复杂的数学公式，使其在文档中更加醒目。

Pandoc 生成 HTML 或 EPUB 需要使用 MathML 或 MathJax，如：

```
pandoc -s --mathjax -o document.epub document.md
```

Pandoc 生成 DOCX 时将使用 OMML 数学标记来呈现：

```
pandoc -s -o document.docx document.md
```

Pandoc 生成 LaTeX 时将逐字显示，周围环绕着`\(...\)`（行内公式）或`\[...\]`（行间公式）。

Pandoc 生成 PDF 时先调用 LaTeX 引擎。

行内公式 Inline math

```
Euler's formula is remarkable: $e^{i\pi} + 1 = 0$.
```

渲染效果如下：

Euler's formula is remarkable: $e^{i\pi} + 1 = 0$.

行间公式 Display Math

```
When $a \ne 0$, there are two solutions to $(ax^2 + bx + c = 0)$ and they are

$$ x = \dfrac{-b \pm \sqrt{b^2-4ac}}{2a} $$
```

[9] https://github.com/jgm/pandoc/issues/2139

渲染效果如下：

When $a \neq 0$, there are two solutions to $(ax^2 + bx + c = 0)$ and they are

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

交叉引用 Cross Referencing

使用 pandoc-crossref

安装 pandoc 和 pandoc-crossref

要想使用 pandoc-crossref，需要安装相互兼容的 pandoc 和 pandoc-crossref；否则 Pandoc 会提示兼容性问题。如：pandoc-3.6.4 和 pandoc-crossref-v0.3.18.2。

- pandoc 下载地址
- pandoc-crossref 下载地址

解压缩 pandoc 可执行文件到一个目录中，如 ~/.local/bin/pandoc。解压缩 pandoc-crossref 可执行文件到 ~/.local/bin/pandoc-crossref 中. 帮助文档拷贝到 ~/.local/share/man/man1 中，目录结构如下：

```
$ tree ~/.local/share/man
/home/USERNAME/.local/share/man
└── man1
    ├── pandoc.1.gz
    ├── pandoc-crossref.1
    ├── pandoc-lua.1.gz
    └── pandoc-server.1.gz
```

运行 pandoc-crossref

首先，需要设置环境变量 PATH：

```
export PATH=~/.local/bin:$PATH
```

在 macOS 中，首次运行 `pandoc --filter pandoc-crossref` 时会提示：“无法打开“pandoc-crossref”，因为无法验证开发者。”点击“取消”，然后在“系统偏好设置-安全性与隐私-通用”中“仍然允许”通过验证。

直接转换成 PDF 时的错误处理

直接转换时出现以下错误:

```
Error producing PDF.
! Undefined control sequence.
l.340 (\passthrough
```

可以先转换成.tex 文件，然后在文件的导言区中加入:

```
\usepackage{listings}
\newcommand{\passthrough}[1]{#1}
```

或者：添加 `-M listings` 选项[10]。

引用章节 Referencing Sections

在文本中定义章节标签 {#sec:label}，然后引用该标签 [@sec:label]:

```
# This is a chapter {#sec:ch}

## This is the first section {#sec:a}

Lorem ipsum dolor sit amet, consectetur adipiscing elit.

## This is the second section {#sec:b}

Sed iaculis pellentesque tempor. Etiam fringilla orci ut est efficitur, vitae
    iaculis odio convallis.

See [@sec:a] and [@sec:b].
```

编译时必须使用 `--number-sections`，如:

```
pandoc --number-sections --filter pandoc-crossref -o output.pdf input.md
```

引用图片 Referencing Figures

在文本中定义图片标签 {#fig:label}，然后引用该标签 [@fig:label]:

```
![alt text](/path/to/image){#fig:label}
```

如果需要加入属性:

```
![alt text](/path/to/image){#fig:label width=32px}
```

[10] pandoc-crossref と Eisvogel を併用する

引用表格 Referencing Tables

在文本中定义表格标签 {#tbl:label}，然后引用该标签 [@tbl:label]：

```
 a   b   c
--- --- ---
 1   2   3
 4   5   6

: Simple Table {#tbl:label}
```

引用公式 Referencing Equations

在文本中定义公式标签 {#eq:label}，然后引用该标签 [@eq:label]：

```
$$ f(x) = x^2 + x + 1 $$ {#eq:label}
```

前缀

单个引用：[@sec:label] 默认前缀是 sec.，显示如：sec. 1.5

合并引用：[@sec:label1; @sec:label2]，显示如：secs. 1.5, 1.6

去掉前缀：[-@sec:label]，显示如：1.5

自定义前缀：

```
---
secPrefix: ["Sect.", "Sects."]
---
```

类型	metadata
章节	secPrefix
图片	figPrefix
表格	tblPrefix
公式	eqnPrefix

文献引用 Citations

Pandoc 支持文献引用，常用格式为 BibTeX 或 BibLaTeX，结合 Markdown 或 LaTeX 进行引用。Pandoc 支持通过 CSL（Citation Style Language）文件自定义引用文献风格，用于格式化参考文献和引文。Pandoc 支持多种文献数据格式，常用的是 BibTeX (.bib) 或 BibLaTeX，也支持 JSON、YAML 等格式。

创建 BibTeX 文件

创建一个名为 `references.bib` 的文件，内容如下：

```
@book{knuth1997,
  author    = {Donald E. Knuth},
  title     = {The Art of Computer Programming, Volume 1: Fundamental
      Algorithms},
  year      = {1997},
  publisher = {Addison-Wesley},
  address   = {Reading, MA}
}

@article{lamport1994,
  author  = {Leslie Lamport},
  title   = {LaTeX: A Document Preparation System},
  journal = {Software: Practice and Experience},
  volume  = {24},
  number  = {3},
  year    = {1994},
  pages   = {307--317}
}
```

CSL 文件

Pandoc 使用 CSL 文件定义引文和参考文献的格式。你可以从 CSL GitHub 仓库 下载现成的 CSL 文件（如 APA、MLA、Chicago 等），或自定义 CSL 文件。例如：

- APA 风格：`apa.csl`
- Chicago 风格：`chicago-author-date.csl`

如果不指定 CSL 文件，Pandoc 默认使用 Chicago 作者-日期格式。

预览参考文献的风格：https://www.zotero.org/styles

以下是一些常见的 CSL 风格及其用途：

- **APA** (`apa.csl`)：心理学、社会科学常用，强调作者和年份。
- **MLA** (`modern-language-association.csl`)：人文学科，注重作者和标题。

- **Chicago** (`chicago-author-date.csl` 或 `chicago-note-bibliography.csl`): 历史学、文学等，分为作者-日期和笔记-书目两种。
- **IEEE** (`ieee.csl`): 工程技术领域，编号引文格式。

CSL 文件存储位置:

- Pandoc 有一个默认的数据目录，用于存储 CSL 文件、模板等资源。可以通过以下命令查看 Pandoc 的数据目录:

  ```
  pandoc --version
  ```

  输出中会显示 `Default user data directory`，通常是:

  - Linux/macOS: `~/.pandoc/` 或 `~/.local/share/pandoc/`
  - Windows: `C:\Users\<用户名>\AppData\Roaming\pandoc\`

- 你可以将 CSL 文件放在这个目录下的 `csl/` 子文件夹中，例如 `~/.pandoc/csl/style.csl`。
- 放在这里后，Pandoc 可以直接识别文件名，无需指定完整路径。例如:

  ```
  pandoc input.md --csl=style.csl -o output.pdf
  ```

使用参考文献

指定.bib 文件和.csl 文件

可以通过命令行选项设置，如:

`--bibliography=references.bib --csl=apa.csl`

或也可以通过 YAML 文件指定:

```
---
bibliography: references.bib
csl: apa.csl
---
```

文献引用语法

语法:

`@knuth1997`

`@lamport1994, p. 310`

渲染效果与 CSL 格式有关。

编译

运行以下 Pandoc 命令，将 Markdown 文件转换为目标格式（如 PDF 或 HTML）：

```
pandoc example.md --citeproc -o example.pdf
```

当与 pandoc-crossref 一起使用时，处理顺序很重要。通常情况下，你需要先运行 pandoc-crossref，然后再运行 --citeproc。这是因为 pandoc-crossref 使用了与 --citeproc 相似的 [@...] 引用语法，先运行 pandoc-crossref 可以避免两者之间的冲突。

```
pandoc example.md --filter pandoc-crossref --citeproc -o example.pdf
```

注意：pandoc-crossref 并不能为参考文献建立链接。

参考资料 References

- Markdown Guide
- Markdown 指南中文版
- A writer's guide to Pandoc's Markdown
- Introducing Markdown and Pandoc