

# CommonMark 规范

版本 0.31.2 (2024-01-28)

约翰·麦克法兰



## [1 简介](#)

[1.1 什么是 Markdown?](#)

[1.2 为什么需要规范?](#)

[1.3 关于本文档](#)

## [2 准备工作](#)

[2.1 人物与台词](#)

[2.2 标签](#)

[2.3 不安全的字符](#)

[2.4 反斜杠转义](#)

[2.5 实体和数字字符引用](#)

## [3 块和内联](#)

[3.1 优先级](#)

[3.2 容器块和叶块](#)

## [4 树叶块](#)

[4.1 主题休息](#)

[4.2 ATX 标题](#)

[4.3 Setext 标题](#)

[4.4 缩进代码块](#)

[4.5 隔离代码块](#)

[4.6 HTML 块](#)

[4.7 链接引用定义](#)

[4.8 段落](#)

[4.9 空行](#)

## [5 容器块](#)

[5.1 区块引用](#)

[5.2 列出项目](#)

[5.3 列表](#)

## [6 内联](#)

[6.1 代码跨度](#)

[6.2 强调和强烈强调](#)

[6.3 链接](#)

[6.4 图像](#)

[6.5 自动链接](#)

[6.6 原始 HTML](#)

[6.7 硬换行](#)

[6.8 软换行](#)

[6.9 文本内容](#)

## [附录：解析策略](#)

[概述](#)

[第一阶段：块结构 第二阶](#)

[段：内联结构](#)

# 1 简介

## 1.1 什么是 Markdown?

Markdown 是一种用于编写结构化文档的纯文本格式，基于电子邮件和 usenet 帖子中指示格式的惯例。它由 John Gruber (在 Aaron Swartz 的帮助下) 开发，并于 2004 年以[语法描述](#) 以及一个 Perl 脚本 (Markdown.pl) 用于将 Markdown 转换为 HTML。在接下来的十年中，许多语言开发了数十种实现。有些扩展了原始的 Markdown 语法，增加了脚注、表格和其他文档元素的约定。有些允许 Markdown 文档以 HTML 以外的格式呈现。Reddit、StackOverflow 和 GitHub 等网站有数百万人使用 Markdown。Markdown 开始在网络之外得到使用，用于创作书籍、文章、幻灯片、信件和讲义。

Markdown 与许多其他轻量级标记语法的区别在于它的可读性，而这些语法通常更容易编写。正如 Gruber 所写：

Markdown 格式语法的首要设计目标是使其尽可能易于阅读。其理念是，Markdown 格式的文档应可按原样以纯文本形式发布，而不像是用标签或格式说明标记的。[\(https://daringfireball.net/projects/markdown/\)](https://daringfireball.net/projects/markdown/)

可以通过比较以下样本来说明这一点AsciiDoc 使用 Markdown 的等效示例。以下是 AsciiDoc 手册中的 AsciiDoc 示例：

1. 列出第一项。

+

列表项一接着第二段，后面跟着一个缩进的块。

+

```
$ mv *.sh ~/tmp
```

..... \$ ls \*.sh

+

列表项继续第三段。

2. 列表项二以开放块的形式继续。

+

--  
本段是前一个列表项的一部分。

a. 此列表是嵌套的，不需要显式的项目延续。

+  
本段是前一个列表项的一部分。

b. 列表项 b。

本段属于外层清单的第二项。

--

以下是 Markdown 中的等效内容：

1. 列出第一项。

列表项一接着第二段，后面跟着一个缩进的块。

```
$ ls *.sh  
$ mv *.sh ~/tmp
```

列表项继续第三段。

2. 列表项二以开放块形式继续。此段落是前一个列表项的一部

分。

1. 此列表是嵌套的，不需要显式项延续。此段落是前一个列表项的一部分。

2. 列出项目 b。

本段属于外层清单的第二项。

AsciiDoc 版本可以说更容易编写。您无需担心缩进。但 Markdown 版本更容易阅读。列表项的嵌套在源代码中显而易见，而不仅仅是在处理后的文档中。

## 1.2 为什么需要规范？

约翰·格鲁伯的[Markdown 语法的规范描述](#)没有明确地指定语法。以下是它没有回答的一些问题示例：

1. 子列表需要缩进多少？规范规定，连续段落需要缩进四个空格，但对子列表没有完全明确规定。人们很自然地认为它们也必须缩进四个空格，但Markdown.pl不需要。这几乎不是一个“极端情况”，并且实际文档中关于这个问题的实现之间的分歧经常会导致用户感到惊讶。（见[John Gruber 的评论](#)。）

2. 引用块或标题前是否需要空行？大多数实现不需要空行。但是，这可能会导致文本硬换行，并导致解析出现歧义（请注意，有些实现将标题放在引用块内，而其他实现则不这样做）。（John Gruber 也曾说过[赞成要求空行](#)。）

3. 缩进的代码块前需要有一个空行吗? (Markdown.pl需要它, 但文档中没有提到, 并且有些实现不需要它。)

段落  
代码?

4. 确定列表项何时被包裹在 < 中的确切规则是什么对标签? 列表可以部分“宽松”而部分“严格”吗? 我们应该如何处理这样的列表?

—

2. —  
3. 三

或者这个?

—  
- 一个

-b  
2. 二

(John Gruber 发表了一些相关评论[这里](#)。

5. 列表标记可以缩进吗? 有序列表标记可以右对齐吗?

8. 第 1 项  
9. 第 2 项  
10. 第2a项

6. 这是一个在第二项中以主题分隔的列表, 还是两个以主题分隔的列表?

\*一个  
\*\*\*\*\*  
\* b

7. 当列表标记由数字变为项目符号时, 我们有两个列表还是一个列表? (Markdown 语法描述建议有两个列表, 但 perl 脚本和许多其他实现只生成一个列表。)

1. 费用  
2. 国际羽联  
- 敌人  
- 熏

8. 内联结构标记的优先规则是什么? 例如, 以下内容是否为有效链接, 或者代码跨度是否优先?

[反引号 (`)](/url) 和 [另一个反引号 (`)](/url)。

9. 强调标记和强调符号的优先顺序规则是什么? 例如, 下面的句子该如何解析?

\* foo \*酒吧\* 巴兹\*

10. 块级和行级结构之间的优先规则是什么? 例如, 以下内容应如何解析?

- ` 较长的代码跨度可以包含这样的连字符  
- 并且它可能会把事情搞砸`

11. 列表项可以包括章节标题吗? (Markdown.pl不允许这样做, 但允许块引用包含标题。)

- # 标题

12. 列表项可以为空吗?

\*一个  
\*

13. 可以在块引用或列表项内定义链接引用吗?

```
> 块引用 [foo].  
>  
> [foo]: /url
```

14. 如果同一引用有多个定义，则哪个定义优先?

```
[foo]: /url1  
[foo]: /url2  
  
[foo][]
```

由于缺乏规范，早期实施者咨询Markdown.pl解决这些歧义。但是 Markdown.pl 存在很多错误，并且在很多情况下会产生明显的糟糕结果，因此它不是一个令人满意的替代规范。

由于没有明确的规范，因此实现方式存在很大差异。因此，用户经常会惊讶地发现，在一个系统（例如 GitHub wiki）上呈现的文档在另一个系统（例如，使用 pandoc 转换为 docbook）上呈现的方式不同。更糟糕的是，由于 Markdown 中没有任何东西算作“语法错误”，因此差异通常不会立即被发现。

## 1.3 关于本文档

本文档试图明确说明 Markdown 语法。它包含许多并列的 Markdown 和 HTML 示例。这些示例旨在兼作一致性测试。随附脚本 spec\_测试.py 可用于针对任何 Markdown 程序运行测试：

```
python 测试/spec_tests.py --spec spec.txt --program 程序
```

由于本文档描述了如何将 Markdown 解析为抽象语法树，因此使用语法树的抽象表示而不是 HTML 是合理的。但 HTML 能够表示我们需要进行的结构区分，并且选择 HTML 进行测试使得我们能够在不编写抽象语法树渲染器的情况下针对实现运行测试。

请注意，规范并未强制要求 HTML 示例的所有功能。例如，规范规定了哪些内容算作链接目标，但并未要求对 URL 中的非 ASCII 字符进行百分比编码。要使用自动测试，实施者需要提供符合规范示例要求的渲染器（对 URL 中的非 ASCII 字符进行百分比编码）。但符合要求的实施可以使用不同的渲染器，并且可以选择不对 URL 中的非 ASCII 字符进行百分比编码。

本文档由文本文件生成，规范.txt，用 Markdown 编写，带有一个用于并行测试的小扩展。脚本工具/makespec.py 可以用来转换规范.txt 转换为 HTML 或 CommonMark（然后可以转换成其他格式）。

在示例中，→字符用于表示制表符。

## 2 准备工作

### 2.1 人物与台词

任意序列人物 是有效的 CommonMark 文档。

一个特点 是 Unicode 代码点。尽管有些代码点（例如组合重音符号）在直观意义上并不对应于字符，但就本规范而言，所有代码点都算作字符。

本规范没有指定编码；它认为行是由人物 而不是字节。符合要求的解析器 可能限于某种编码。

一个线 是零个或多个序列人物 除了换行符 (U+000A) 或回车符 (U+000D) 随后是行结束符 或文件末尾。

一个行结束符 是换行符 (U+000A) 回车符 (U+000D) 后面没有换行符，或者没有回车符和换行符。

不包含任何字符的行，或者仅包含空格的行 (U+0020) 或标签 (U+0009) 被称为空行。

本规范中将使用以下字符类的定义：

一个Unicode 空白字符是 Unicode 中的一个字符扎斯常规类别，或选项卡 (U+0009) 换行符 (U+000A) 换页符 (U+000C) 或回车符 (请参阅“U+000D”一文)。

Unicode 空格是一个或多个序列Unicode 空格字符。

一个标签是U+0009。

一个空间是U+0020。

一个ASCII 控制字符是介于U+0000–1F (包括) 或U+007F。

一个ASCII 标点符号是!、"、#、\$、%、&、'、(、)、\*、+、,、-、.、/ (U+0021-2F)、:、;、<、=、>、?、@ (U+003A-0040)、[、\、]、^、\_、` (U+005B-0060)、{、|、}或~ (U+007B-007E)。

一个Unicode 标点符号是 Unicode 中的一个字符磷 (标点符号) 或小号 (符号 (symbol)) 的一般类别。

## 2.2 标签

行中的制表符不会扩展为空格。然而，在空格有助于定义块结构的上下文中，制表符的行为就像它们被具有 4 个字符的制表位的空格替换一样。

因此，例如，在缩进的代码块中，可以使用制表符代替四个空格。（但请注意，内部制表符将作为文字制表符传递，而不是扩展为空格。）

### 示例 1

→ foo→baz→→bim

```
<pre><code>foo→baz→→bim</code></pre>
```

### 示例 2

• → foo→baz→→bim

```
<pre><code>foo→baz→→bim</code></pre>
```

### 示例 3

• . . → a → a  
. . . Ú → a

```
<pre><code>a→a  
Ú→a</code></pre>
```

在以下示例中，列表项的延续段落使用制表符缩进；这与使用四个空格缩进的效果完全相同：

### 示例 4

• - · f 哦

```
<ul>  
<li>  
<p>foo</p>  
<p>酒吧</p>  
</li>  
</ul>
```

### 示例 5

- 富  
→ → 酒吧

```
<ul>  
<li>  
<p>foo</p>  
<pre><code>.. 酒吧</code></pre>  
</li>  
</ul>
```

通常，块引用开头的 > 后面可以随意跟一个空格，但空格不被视为内容的一部分。在以下情况下，> 后面跟一个制表符，它被视为扩展为三个空格。由于其中一个空格被视为分隔符的一部分，富被认为是在块引用上下文中缩进六个空格，因此我们得到一个以两个空格开头的缩进代码块。

### 示例 6

> → → foo

```
<blockquote>
<pre><代码>· · 富
</代码> </ pre>
</blockquote>
```

## 示例 7

- → → foo

```
<ul>
<li>
<pre><代码>· · 富
</代码> </ pre>
</li>
</ul>
```

## 示例 8

· · 富  
→ 酒吧

```
<pre><code>foo
酒吧
</code> </ pre>
```

## 示例 9

· · 哟  
· · ~ 酒吧  
→ ~ 巴兹

```
<ul>
<li>foo
<ul>
<li>酒吧
<ul>
<li>巴兹</li>
</ul>
</li>
</ul>
</li>
</ul>
```

## 示例 10

# → Foo

```
<h1>Foo</h1>
```

## 示例 11

\* → \* → \* →

```
</小时>·
```

## 2.3 不安全的字符

出于安全原因， Unicode 字符U+0000必须用替换字符 ( (U+FFFD) ) 。

## 2.4 反斜杠转义

任何 ASCII 标点符号都可以用反斜杠转义：

### 示例 12

!\"#\$%&`(\")\*+,,-.\\:;|<|=|>|?|@\\[\\]\\^\\\_\\{|~

```
<p>!#$%&'()*+,-./;=>?@[\\]^_`{}|~ </p>
```

其他字符前的反斜杠将被视为文字反斜杠：

### 示例 13

\→|A|a| · |3|φ|«

```
<p>\→|A|a| · |3|φ|« </p>
```

转义字符将被视为常规字符，不具有其通常的 Markdown 含义：

### 示例 14

*不是强调*	<p>*不·强调*
\  · 不是—标签	  · 不是—标签
\[不是 —链接](/foo)	[不是 —链接](/foo)
\`不·代码`	`不·代码`
1\. ·不是—列表	1. ·不是—列表
\* ·不是—列表	* ·不是—列表
\# ·不是—标题	# ·不是—标题
\[foo]: · /网址 ·“不是—参考”	[foo]: · /网址 ·“不· —参考”
\ö · 不是—特点 · 实体	&ouml; · 不是—特点 · 实体</p>

如果反斜杠本身被转义，则以下字符不会转义：

#### 示例 15

\\"\*强调\*

<p>\<em>强调</em></p>

行尾的反斜杠是硬换行：

#### 示例 16

foo\  
酒吧

<p>foo<br · />  
</p>

反斜杠转义在代码块、代码跨度、自动链接或原始 HTML 中不起作用：

#### 示例 17

\` ` \| \` · `

<p><code>\` ` \| \` </code></p>

#### 示例 18

\ . . \| ]

<pre><code>\ . . \| ]</code></pre>

#### 示例 19

~~~  
\[]  
~~~

<pre><code>\[]</code></pre>

#### 示例 20

<https://example.com?find=\\*>

<p><a href="https://example.com?  
find=%5C%">https://example.com?find=\\*</a></p>

#### 示例 21

<—href="/bar\/">

<—href="/bar\/">

但它们在所有其他情况下都有效，包括 URL 和链接标题、链接引用和信息字符串 在隔离代码块：

#### 示例 22

[foo](/bar\\* · “标题” )

<p><a href="/栏\*" · title="ti\*tle">foo</a></p>

#### 示例 23

[foo]

<p><a href="/栏\*" · title="ti\*tle">foo</a></p>

[foo]: · /酒吧\\* · “标题”

#### 示例 24

\` ` ` foo\+bar  
富、

<pre><code>·类=“语言-foo + bar” > foo  
</code></pre>

有效的 HTML 实体引用和数字字符引用可用于代替相应的 Unicode 字符，但以下情况除外：

- 代码块和代码跨度中无法识别实体和字符引用。
- 实体和字符引用不能代替 CommonMark 中定义结构元素的特殊字符。例如，尽管 42；可以代替文字 \* 字符，\* 不能代替强调分隔符、项目符号列表标记或主题分隔符中的 \*。

符合 CommonMark 规范的解析器不需要存储有关特定字符在源中是使用 Unicode 字符还是实体引用来表示的信息。

**实体引用** 由 & + 任何有效的 HTML5 实体名称 + ; 组成。文档 <https://html.spec.whatwg.org/entities.json> 用作有效实体引用及其对应代码点的权威来源。

### 示例 25

• & · & 复制; ◆ AElig; ◆ D  
¾ · & Hilbert 空间; · & 差分 D;  
& 顺时针轮廓积分; · ≥ /

<p> • & · © AE D  
¾ H 空间  
ƒ · ≥ / 对

**十进制数字字符引用** 由 + 1-7 个阿拉伯数字 + ; 组成。数字字符引用被解析为相应的 Unicode 字符。无效的 Unicode 代码点将由替换字符 ( (U+FFFD) ) 替换。出于安全原因，代码点 U+0000 也将被取代 U+FFFD。

### 示例 26

# · Ä · 𩶲 ·  
<p># 噢 编 ◆ </p>

**十六进制数字字符引用** 由 + 组成十或者 x +1-6 个十六进制数字 + ; 的字符串。它们也被解析为相应的 Unicode 字符（这次用十六进制数字而不是十进制数字指定）。

### 示例 27

" · அ · இ  
<p> " · அ இ </p>

以下是一些不存在的东西：

### 示例 28

· &x; .. ◆  
粳  
& abcdef0;  
这尚未定义; · & 你好? ;  
<p>&nbsp · &x; .. &#amp;#; ◆  
粳  
& abcdef0;  
这尚未定义; · & 嗨?;</p>

尽管 HTML5 确实接受一些不带尾随分号的实体引用（例如 & 复制），这里无法识别这些，因为这会使语法变得太模糊：

### 示例 29

& 复制  
<p>&复制</p>

HTML5 命名实体列表中不存在的字符串也不能被识别为实体引用：

### 示例 30

& MadeUpEntity;  
<p>&MadeUpEntity;</p>

实体和数字字符引用可在除代码跨度或代码块之外的任何上下文中识别，包括 URL、[链接标题](#)，和[隔离代码块信息字符串](#)：

### 示例 31

<— href="öö.html"> <— href="öö.html">

### 示例 32

```
[foo](/fÃouml;Ãouml; · “fÃrÃuml;örÃuml;” )  
<p><a href="/f%C3%B6%C3%B6" · title="fÃöö">foo</a>  
</p>
```

### 示例 33

```
[foo]  
<p><a href="/f%C3%B6%C3%B6" · title="fÃöö">foo</a>  
</p>  
[foo]: ·/fÃöö · “fÃrÃuml;örÃuml;”
```

### 示例 34

```
``` 弗朗西斯  
富、  
  
<pre><代码> · class="language-fÃöö">foo  
</代码> </ pre>
```

实体和数字字符引用在代码跨度和代码块中被视为文字文本：

### 示例 35

```
```  
  
<p><code>f&ouml;öuml;</code></p>
```

### 示例 36

```
. . . 碰碰  
  
<pre><code>f&ouml;öuml;</code></pre>
```

实体和数字字符引用不能代替 CommonMark 文档中表示结构的符号。

### 示例 37

```
*foo*  
* foo *  
  
<p>*foo*  
<em>foo</em></p>
```

### 示例 38

```
* . 富  
* 富  
  
<p>* </p>  
<ul>  
<li>foo</li>  
</ul>
```

### 示例 39

```
foo bar  
  
<p>foo  
</p>
```

### 示例 40

```
foo  
  
<p>→foo</p>
```

### 示例 41

```
[a](网址 · “奶头”  
  
<p>[a](网址 · " tit") </p>
```

---

## 3 块和内联

我们可以将文档视为一系列**块** — 结构元素，如段落、区块引用、列表、标题、规则和代码块。一些区块（如区块引用和列表项）包含其他区块；其他区块（如标题和段落）包含**排队** 内容——文本、链接、强调文本、图像、代码跨度等等。

### 3.1 优先级

块结构指示符始终优先于内联结构指示符。因此，例如，以下是一个包含两个项目的列表，而不是一个包含代码跨度的项目的列表：

### 示例 42

```
<ul>
<li>`一个</li>
<li>二`</li>
</ul>
```

这意味着解析可以分两步进行：首先，可以辨别文档的块结构；其次，可以解析段落、标题和其他块构造中的文本行以获取内联结构。第二步需要有关链接引用定义的信息，这些信息仅在第一步结束时可用。请注意，第一步需要按顺序处理行，但第二步可以并行化，因为一个块元素的内联解析不会影响任何其他块元素的内联解析。

## 3.2 容器块和叶块

我们可以将块分为两种类型：[容器块](#)，其中可以包含其他块，并且[叶块](#)，但不能。

---

## 4 树叶块

本节介绍构成 Markdown 文档的不同类型的叶块。

### 4.1 主题休息

一行由最多三个可选的缩进空格组成，后跟三个或更多匹配的 -、\_ 或 \* 字符序列，每个字符后跟任意数量的空格或制表符，形成[专题休息](#)。

#### [示例 43](#)

```
*** <小时>*
--- <小时>*
____ <小时>*
```

错误字符：

#### [示例 44](#)

```
+++ <p>+++</p>
```

#### [示例 45](#)

```
==== <p>====</p>
```

字符数不足：

#### [示例 46](#)

```
-- <p>--*
** <* >*
____
```

最多允许三个缩进空格：

#### [示例 47](#)

```
* ** <小时>*
* * * <小时>*
.. * ** <小时>*
```

四个空格的缩进太多了：

#### [示例 48](#)

```
... *** <pre><代码>***</代码></pre>
```

#### [示例 49](#)

福

...\*\*\*

```
<p>Foo  
* * *</p>
```

可以使用三个以上的字符：

#### 示例 50

```
-----  
<小时>.
```

字符之间允许有空格和制表符：

#### 示例 51

```
-----  
<小时>.
```

#### 示例 52

```
* * . * . * * * * * *
```

```
<小时>.
```

#### 示例 53

```
-----
```

```
<小时>.
```

末尾允许使用空格和制表符：

#### 示例 54

```
-----
```

```
<小时>.
```

但是，此行中不能出现其他字符：

#### 示例 55

```
- - - - -一个  
一个---  
---一个---
```

```
<p> - - - -</p>  
<p>一个-----</p>  
<p>---一个---</p>
```

要求除空格或制表符之外的所有字符都相同。因此，这不是主题中断：

#### 示例 56

```
* - *
```

```
<p><em>-</em></p>
```

主题分隔符前后不需要空行：

#### 示例 57

```
- 富  
***  
- 酒吧
```

```
<ul>  
<li>foo</li>  
</ul>  
<hour>  
<ul>  
<li>酒吧</li>  
</ul>
```

主题停顿可以打断一个段落：

#### 示例 58

```
福  
***  
酒吧
```

```
<p>食物</p>  
<hour>  
<p>酒吧</p>
```

如果满足上述条件的一条虚线可以解释为主题断点，那么它也可以解释为[设置文本标题](#)，解释为[设置文本标题](#)优先。因此，例如，这是一个setext 标题，而不是后面跟着主题分隔符的段落：

## [示例 59](#)

福  
---  
酒吧

```
<h2>Foo</h2>
<p>酒吧</p>
```

当主题分隔符和列表项都是一行的可能解释时，主题分隔符优先：

## [示例 60](#)

\* 福  
\* \* \*  
\* 酒吧

```
<ul>
<li>Foo</li>
</ul>
<小时>*
<ul>
<li>酒吧</li>
</ul>
```

如果要在列表项中按主题划分，请使用不同的项目符号：

## [示例 61](#)

- 福  
- \* \* \*

```
<ul>
<li>Foo</li>
<里>
<小时>*
</li>
</ul>
```

## 4.2 ATX 标题

一个[ATX 标题](#)由一串字符组成，解析为内联内容，位于 1 至 6 个未转义的 # 字符的开始序列和任意数量的未转义 # 字符的可选结束序列之间。# 字符的开始序列后面必须跟空格或制表符，或者跟在行尾。可选的结束序列 # 前面必须跟空格或制表符，并且后面只能跟空格或制表符。开始 # 字符前面最多可以跟三个缩进空格。标题的原始内容在解析为内联内容之前会删除前导和尾随空格或制表符。标题级别等于开始序列中的 # 字符数。

简单标题：

## [示例 62](#)

# 富  
# # 富  
# ## 富  
##### 富  
##### 富  
##### 富

```
<h1>foo</h1>
<h2>foo</h2>
<h3>foo</h3>
<h4>foo</h4>
<h5>foo</h5>
<h6>foo</h6>
```

超过六个 # 字符不是标题：

## [示例 63](#)

##### 富

```
<p>##### 富</p>
```

除非标题为空，否则 # 字符与标题内容之间至少需要一个空格或制表符。请注意，目前许多实现不需要空格。但是，空格是[原始 ATX 实现](#)，它有助于防止以下内容被解析为标题：

## [示例 64](#)

```
# 5 螺栓  
# 并号
```

```
<p>#5 螺栓</p>  
<p># 标签</p>
```

这不是一个标题，因为第一个 # 被转义了：

### 示例 65

```
\## 富
```

```
<p>## </p>
```

内容被解析为内联：

### 示例 66

```
# 富 . * 酒吧* .\*巴兹\*
```

```
<h1>foo .<em>酒吧</em> .*巴兹*</h1>
```

解析内联内容时，前导和尾随空格或制表符会被忽略：

### 示例 67

```
# .....富.....
```

```
<h1>foo</h1>
```

最多允许三个缩进空格：

### 示例 68

```
# ## f哦  
. # # 富  
. . # 富
```

```
<h3>foo</h3>  
<h2>foo</h2>  
<h1>foo</h1>
```

四个空格的缩进太多了：

### 示例 69

```
. . .#. f哦
```

```
<pre><code>#. 富  
</代码></pre>
```

### 示例 70

```
富  
. . #. b应收账款
```

```
<p>foo  
# </p>
```

结束序列 # 字符是可选的：

### 示例 71

```
# ## 富 . # #  
. # # # . . 酒吧. . . . # # #
```

```
<h2>foo</h2>  
<h3>酒吧</h3>
```

它不需要与开头序列的长度相同：

### 示例 72

```
# 富 . #####  
##### 富 . # #
```

```
<h1>foo</h1>  
<h5>foo</h5>
```

结束序列后允许使用空格或制表符：

### 示例 73

```
### 富 . ##### . . .
```

```
<h3>foo</h3>
```

一串 # 字符，后面除了空格或制表符之外的任何内容都不是结束序列，但算作标题内容的一部分：

### 示例 74

```
### 富。### b
```

```
<h3>foo •# # # *;/h3></h3>
```

结束序列前面必须有一个空格或制表符：

#### 示例 75

```
# foo#
```

```
<h1>foo#</h1>
```

反斜杠转义的 # 字符不算作结束序列的一部分：

#### 示例 76

```
### 富。\\###  
# 富。# \\##  
# 富。\\#
```

```
<h3>foo •### ### </h3>  
<h2>foo •### ### </h2>  
<h1>foo •# </h1>
```

ATX 标题不需要用空行与周围内容分隔，并且可以打断段落：

#### 示例 77

```
****  
# # 富  
*****
```

```
<小时>•  
<h2>foo</h2>  
<小时>•
```

#### 示例 78

```
福。酒吧  
# 巴兹  
酒吧富
```

```
<p>Foo •</p>  
<h1>巴兹</h1>  
<p>酒吧 •</p>
```

ATX 标题可以为空：

#### 示例 79

```
# #•  
#  
### ####
```

```
<h2></h2>  
<h1></h1>  
<h3></h3>
```

### 4.3 Setext 标题

一个[设置文本标题](#)由一行或多行文本组成，中间不留空行，第一行缩进不超过 3 个空格，后跟[setext 标题下划线](#)。文本行必须如此，如果它们后面没有 setext 标题下划线，它们将被解释为一个段落：它们不能被解释为[代码围栏](#)，[ATX 标题](#)，[区块引用](#)，[专题休息](#)，[列表项](#)，或者[HTML 块](#)。

一个[setext 标题下划线](#)是一系列 = 字符或一系列 - 字符，缩进不超过 3 个空格且尾随空格或制表符任意数量。

如果标题中使用 = 字符，则该标题为 1 级标题[setext 标题下划线](#)，如果使用 - 字符，则为 2 级标题。标题的内容是将前面的文本行解析为 CommonMark 内联内容的结果。

一般情况下，setext 标题前后不需要空行，但由于它不能打断段落，所以当 setext 标题位于段落之后时，两者之间需要有一个空行。

简单示例：

#### 示例 80

```
福。* 酒吧*  
=====
```

```
<h1>Foo •<em>酒吧</em></h1>  
<h2>Foo •<em>酒吧</em></h2>
```

```
福。* 酒吧*  
-----
```

标题内容可能跨越多行：

## [示例 81](#)

福 • \* 酒吧  
巴茲\*  
====

```
<h1>Foo •<em>酒吧
</em><h1>巴茲</em>
```

内容是将标题的原始内容解析为内联内容的结果。标题的原始内容是通过连接行并删除首尾空格或制表符形成的。

## [示例 82](#)

• 福 • 酒吧  
巴茲\*→  
====

```
<h1>Foo •<em>酒吧
</em><h1>巴茲</em>
```

下划线可以是任意长度：

## [示例 83](#)

福

```
-->
<h2>Foo</h2>
<h1>Foo</h1>
```

福  
=

标题内容前最多可以有三个空格的缩进，并且不需要与下划线对齐：

## [示例 84](#)

. . 福  
---  
.福  
----  
.福  
. ===

```
<h2>Foo</h2>
<h2>Foo</h2>
<h1>Foo</h1>
```

四个空格的缩进太多了：

## [示例 85](#)

. . . 福  
. . . ---  
. . . 福  
---  
福  
</代码> </ pre>  
<小时>•

```
<pre><code>Foo
-->
```

setext 标题下划线前面最多可以有三个缩进空格，并且可以有尾随空格或制表符：

## [示例 86](#)

福  
. . . . .

```
<h2>Foo</h2>
```

四个空格的缩进太多了：

## [示例 87](#)

福  
. . . .

```
<p>Foo
---</p>
```

setext 标题下划线不能包含内部空格或制表符：

## [示例 88](#)

福  
= =  
福  
---

```
<p>Foo<br/>=</p><p>食物</p></小时>.
```

内容行中的尾随空格或制表符不会导致硬换行：

#### 示例 89

福 . .  
---

```
<h2>Foo</h2>
```

末尾也没有反斜杠：

#### 示例 90

Foo\  
---

```
<h2>Foo\</h2>
```

由于块结构指示符优先于内联结构指示符，因此以下是 setext 标题：

#### 示例 91

` Foo  
---  
`  
<—标题=" . 很多  
---  
的 破折号" />  
  
<h2>` Foo</h2>  
<p>` </p>  
<h2><a . 标题= “ . 很多</h2>  
<p>的 ·破折号"/>/></p>

setext 标题下划线不能是 惰性延续行 在列表项或块引用中：

#### 示例 92

> 福  
---  
>富  
酒吧  
====

```
<blockquote>  
<p>食物</p>  
</blockquote>  
</小时>.
```

#### 示例 93

>富  
酒吧  
====  
  
<blockquote>  
<p>foo  
酒吧  
===</p>  
</blockquote>

#### 示例 94

- 福  
---  
- 福  
酒吧  
====  
  
<ul>  
<li>Foo</li>  
</ul>  
</小时>.

段落和其后的 setext 标题之间需要一个空行，否则段落将成为标题内容的一部分：

#### 示例 95

福  
酒吧  
---  
  
<h2>Foo  
</h2> 酒吧

但一般来说，setext 标题之前或之后不需要空行：

#### 示例 96

---  
福  
---  
酒吧  
---  
巴兹

<小时>  
<h2>Foo</h2>  
<h2>酒吧</h2>  
<p>巴兹</p>

Setext 标题不能为空：

### [示例 97](#)

<p>=====</p>

=====

Setext 标题文本行不得被解释为除段落之外的块结构。因此，以下示例中的破折号行被解释为主题中断：

### [示例 98](#)

---  
---  
<小时>  
<小时>

### [示例 99](#)

- 富  
----  
<ul>  
<li>foo</li>  
</ul>  
<小时>

### [示例 100](#)

. . 富  
---  
<pre><code>foo  
</代码> </ pre>  
<小时>

### [示例 101](#)

> 富  
----  
<blockquote>  
<p>foo</p>  
</blockquote>  
<小时>

如果您想要带有 > 的标题富作为其文字，您可以使用反斜杠转义：

### [示例 102](#)

\> \*富  
----  
<h2>> . </h2> 复制代码

**兼容性说明：**目前大多数 Markdown 实现都不允许 setext 标题的文本跨越多行。但对于如何解释 setext 标题，目前还没有达成共识。

福  
酒吧  
---  
巴兹

人们可以找到四种不同的解释：

1. 段落 “Foo” , 标题 “bar” , 段落 “baz”
2. 段落 “Foo bar” , 主题休息, 段落 “baz”
3. 段落 “Foo bar — baz”
4. 标题 “Foo bar” , 段落 “baz”

我们发现解释 4 最为自然，解释 4 通过允许多行标题增强了 CommonMark 的表达能力。想要解释 1 的作者可以在第一段后放置一个空行：

### [示例 103](#)

福  
酒吧  
---  
巴兹

```
<p>食物</p>
<h2>酒吧</h2>
<p>巴兹</p>
```

想要解释 2 的作者可以在主题分隔符周围放置空行，

#### [示例 104](#)

福  
酒吧  
---  
巴兹

```
<p>Foo
</p>
</小时>。
<p>巴兹</p>
```

或者使用不能算作[setext 标题下划线](#)， 例如

#### [示例 105](#)

福  
酒吧  
\* \* \*
巴兹

```
<p>Foo
</p>
</小时>。
<p>巴兹</p>
```

想要解释 3 的作者可以使用反斜杠转义：

#### [示例 106](#)

福  
酒吧  
\\---  
巴兹

```
<p>Foo
酒吧
---
</p>
```

## 4.4 缩进代码块

一个[缩进的代码块](#)由一个或多个组成[缩进块](#)用空行分隔。[缩进块](#)是一系列非空白行，每行前面都有四个或更多的缩进空格。代码块的内容是行的文字内容，包括尾随[行尾](#)，减去四个空格的缩进。缩进的代码块没有[信息字符串](#)。

缩进的代码块不能打断段落，因此段落和后面的缩进代码块之间必须有一个空行。（但是，代码块和后面的段落之间不需要空行。）

#### [示例 107](#)

```
...一·实现
....缩进· 代码·堵塞
...·简单的
...·缩进· 代码·堵塞
</代码> </ pre>
```

如果对缩进作为代码块的解释与表示材料属于某个[列表项](#)，列表项解释优先：

#### [示例 108](#)

```
...·哦
...酒吧
...<ul>
...<li>
...<p>foo</p>
...<p>酒吧</p>
...</li>
...</ul>
```

#### [示例 109](#)

1. . 富

. . . - . b 应收账

```
<ol>
<li>
<p>foo</p>
<ul>
<li>酒吧</li>
</ul>
</li>
</ol>
```

代码块的内容是文字文本，不会被解析为 Markdown：

#### 示例 110

```
. . . <— />
. . . *你好*
. . . - o 恩
<pre><code><a/>;
* 你好*
- —
</code> </ pre>
```

这里我们有三个由空行分隔的块：

#### 示例 111

```
. . 块1
. . 块2
. .
. .
. . 块3
块1
块2
块3
</代码> </ pre>
```

缩进四个空格以外的任何初始空格或制表符都将包含在内容中，即使在内部空白行中也是如此：

#### 示例 112

```
. . 块1
. .
. . . 块2
块1
.
块2
</代码> </ pre>
```

缩进的代码块不能中断段落。（这允许悬挂缩进等。）

#### 示例 113

```
福
. . 酒吧
<p>Foo
</p>
```

但是，任何非空行，如果缩进少于 4 个空格，代码块就会立即结束。因此，段落可能紧接着缩进的代码：

#### 示例 114

```
. . 富
酒吧
<pre><code>foo
</code> </ pre>
<p>酒吧</p>
```

缩进的代码可以出现在其他类型的块之前或之后：

#### 示例 115

```
# 标题
. . 富
标题
-----
. . 富
-----
<h1>标题</h1>
<pre><code>foo
</code> </ pre>
<h2>标题</h2>
<pre><code>foo
</code> </ pre>
</小时>*
```

第一行前面可以有四个以上的缩进空格：

#### [示例 116](#)

....富  
...酒吧

```
<pre><代码>....富  
酒吧  
</代码></pre>
```

缩进代码块前面或后面的空行不包含在内：

#### [示例 117](#)

....  
...富  
....

```
<pre><code>foo  
</代码></pre>
```

代码块的内容中包含尾随空格或制表符：

#### [示例 118](#)

...富 ..

```
<pre><code>foo ..  
</代码></pre>
```

## 4.5 隔离代码块

一个[代码围栏](#)是至少三个连续的反引号 (`) 或波浪符号 (~) 的序列。（波浪符号和反引号不能混合使用。）[隔离代码块](#)以代码栅栏开始，前面最多有三个缩进空格。

带有起始代码栅栏的行可以选择性地包含代码栅栏后面的一些文本；这些文本被修剪掉前导和尾随空格或制表符，并称为[信息字符串](#)。如果[信息字符串](#)位于反引号栅栏之后，则不得包含任何反引号字符。（此限制的原因是，否则某些内联代码将被错误地解释为栅栏代码块的开头。）

代码块的内容包括所有后续行，直到结束[代码围栏](#)与代码块开头的类型相同（反引号或波浪号），并且反引号或波浪号的数量至少与开头代码栏相同。如果开头代码栏前面有 N 个缩进空格，则每行内容（如果存在）最多会删除 N 个缩进空格。（如果内容行未缩进，则保持不变。如果缩进 N 个或更少，则删除所有缩进。）

结束代码栅栏前面最多可以有三个缩进空格，后面只能是空格或制表符，这些将被忽略。如果到达包含块（或文档）的末尾，但未找到结束代码栅栏，则代码块包含开始代码栅栏之后直到包含块（或文档）末尾的所有行。（另一种规范要求在未找到结束代码栅栏的情况下进行回溯。但这会使解析效率大大降低，并且这里描述的行为似乎没有真正的缺点。）

围栏代码块可以中断一个段落，并且之前或之后不需要空行。

代码围栏的内容将被视为文字文本，而不是内联文本。[信息字符串](#)通常用于指定代码示例的语言，并在班级的属性代码标签。但是，本规范并不要求对[信息字符串](#)。

这是一个带反引号的简单示例：

#### [示例 119](#)

```  
<  
>  
```

```
<pre><code><  
>  
</代码></pre>
```

带有波浪号：

#### [示例 120](#)

```
<pre><code><
>
</code> </pre>
```

少于三个反引号是不够的：

#### 示例 121

```
```
富`
```

```
<p><code>foo</code></p>
```

关闭代码栅栏必须使用与打开栅栏相同的字符：

#### 示例 122

```
```  
AAA  
~~~  
```
```

```
<pre><code>aaa  
~~~  
</code> </pre>
```

#### 示例 123

```
~~~  
AAA  
```  
~~~
```

```
<pre><code>aaa  
```  
</code> </pre>
```

关闭代码围栏必须至少与打开围栏一样长：

#### 示例 124

```
````  
AAA  
````  
``````
```

```
<pre><code>aaa  
````  
</code> </pre>
```

#### 示例 125

```
~~~~  
AAA  
~~~  
~~~~
```

```
<pre><code>aaa  
~~~  
</code> </pre>
```

未封闭的代码块在文档末尾关闭（或封闭的[区块引用](#)或者[列表项](#)）：

#### 示例 126

```
````  
``````
```

```
<pre><code></code></pre>
```

#### 示例 127

```
``````  
````  
AAA  
````
```

```
<pre><code>  
````  
AAA  
</code> </pre>
```

#### 示例 128

```
> . ``  
> AAA  
bbb
```

```
<blockquote>  
<pre><code>aaa  
</code> </pre>  
</blockquote>  
<p>bbb</p>
```

代码块可以全部以空行作为其内容：

#### 示例 129

```
<pre><代码>
    .
    .
</代码> </pre>
```

代码块可以为空：

### 示例 130

```
<pre><代码></代码></pre>
```

栅栏可以缩进。如果开头的栅栏缩进，则内容行将删除等效的开头缩进（如果存在）：

### 示例 131

```
• ``  
AAA  
AAA  
```
```

### 示例 132

```
...\\
AAA
• AAA
AAA
...
...\\
```

<pre><code>aaa  
AAA  
AAA  
</code></pre>

### 示例 133

```
...:  
. . AAA  
. . . AAA  
• AAA  
...:  
  
<pre><code>aaa  
AAA  
AAA  
</code> </ pre>
```

四个空格的缩进太多了：

### 示例 134

```
...`  
...AAA  
...`  
  
<pre><code>````  
AAA  
`  
</code></pre>
```

结束栅栏前最多可以有 3 个缩进空格，并且它们的缩进不需要与开始栅栏的缩进一致：

### 示例 135

```
<`  
AAA  
`>
```

### 示例 136

```
...`  
AAA  
...`  
  
<pre><code>aaa  
</code> </pre>
```

这不是一个结束栅栏，因为它缩进 4 个空格：

### 示例 137

```
<pre><code>aaa  
....`  
</代码> </ pre>
```

代码栅栏（打开和关闭）不能包含内部空格或制表符：

### 示例 138

```
``` .  
AAA
```

```
<p><代码> ·</code>  
啊</p>
```

### 示例 139

```
~~~~~  
AAA  
~~~ ~~
```

```
<pre><code>aaa  
~~~ ~~  
</代码> </ pre>
```

围栏代码块可以打断段落，也可以直接跟在段落后面，中间不需要空行：

### 示例 140

```
富、  
酒吧  
```  
巴兹
```

```
<p>foo</p>  
<pre><code>条形码  
</代码> </ pre>  
<p>巴兹</p>
```

其他块也可以出现在隔离代码块之前或之后，中间不设空行：

### 示例 141

```
富  
---  
~~~  
酒吧  
~~~  
# 巴兹
```

```
<h2>foo</h2>  
<pre><code>条形码  
</代码> </ pre>  
<h1>巴兹</h1>
```

一个信息字符串可以在打开的代码栏后提供。虽然本规范不要求对信息字符串进行任何特殊处理，但第一个单词通常用于指定代码块的语言。在 HTML 输出中，语言通常通过向代码元素由...组成语言-后跟语言名称。

### 示例 142

```
``` ruby  
定义 foo (x)  
· 返回 · 3  
结尾  
```
```

```
<pre><代码> ·类= “language-ruby” >def · foo (x)  
· 返回 · 3  
结尾  
</代码> </ pre>
```

### 示例 143

```
~~~~. . . . 红宝石起始线=3 · $%@#$  
定义 foo (x)  
· 返回 · 3  
结尾  
~~~~~
```

```
<pre><代码> ·类= “language-ruby” >def · foo (x)  
· 返回 · 3  
结尾  
</代码> </ pre>
```

### 示例 144

```
```;
```

```
<pre><代码> ·class="language-;"></code></pre>
```

信息字符串 对于反引号代码块不能包含反引号：

### 示例 145

```
``` 氨基酸 ·  
富
```

```
<p><code>AA</code>  
</p>
```

信息字符串 对于波浪符号代码块，可以包含反引号和波浪符号：

### 示例 146

关闭代码围栏不能有信息字符串：

#### [示例 147](#)

```
```  
``` AAA  
```
```

## 4.6 HTML 块

一个HTML 块是一组被视为原始 HTML 的行（并且不会在 HTML 输出中转义）。

有七种HTML 块，可以通过其开始和结束条件来定义。块以满足开始条件（最多可缩进三个空格）。它以第一个满足匹配条件的后续行结束结束条件或文档的最后一行，或容器块包含当前 HTML 块，如果没有遇到满足结束条件。如果第一行同时满足开始条件和结束条件，该块将只包含该行。

**1.启动条件：**行以字符串<开头前，<脚本，<样式，或<文本区域（您可以使用任何文本（例如，一个英文字符，不区分大小写）后面跟一个空格、制表符、字符串>或行尾。

**结束条件：**行包含结束标记</前>，</script>，</style>，或</文本区域>（不区分大小写；不需要匹配开始标记）。

**2.启动条件：**行以字符串<!-- 开头。**结束条件：**行包含字符串-->。

**3.启动条件：**行以字符串<?开头。**结束条件：**行包含字符串?>。

**4.启动条件：**行以字符串<!开头，后跟一个 ASCII 字母。**结束条件：**行包含字符>。

**5.启动条件：**行以字符串<![开头CDATA[。**结束条件：**行包含字符串]]>。

**6.启动条件：**行以字符串<或</开头，后跟其中一个字符串（不区分大小写）地址、文章、旁边、基础、basefont、blockquote、正文、标题、中心、col、colgroup、dd、详细信息、对话框、dir、div、dl、dt、fieldset、figcaption、图形、页脚、表格、框架、框架集、h1、h2、h3、h4、h5、h6、头部、页眉、hr、html、iframe、图例、li、链接、主页、菜单、menuitem、nav、noframes、ol、optgroup、选项、p、参数、搜索、部分、摘要、表格、tbody、td、tfoot、th、thead、标题、tr、轨道、ul、后跟空格、制表符、行尾、字符串>或字符串/>。

**结束条件：**行后跟空行。

**7.启动条件：**行以完整开始打开标签（与任何标签名称以外前期，脚本，风格，或者文本区域）或完整的结束标签，后跟零个或多个空格和制表符，后跟行尾。

**结束条件：**行后跟空行。

HTML 块持续存在，直到被相应的结束条件或文档的最后一行或其他容器块。这意味着任何 HTML 在 HTML 块内否则可能会被识别为起始条件的条件将被解析器忽略并按原样传递，而不会改变解析器的状态。

例如，<上一篇>在以<开头的 HTML 块内表格>不会影响解析器状态；因为 HTML 块是由启动条件 6 启动的，所以它将在任何空白行结束。这可能会令人惊讶：

#### [示例 148](#)

```
<表格><tr><td>
<前>
** 你好**,
世界_。
</pre>
</td> </tr> </table>

<表格><tr><td>
<前>
** 你好**,
<p><em>世界</em>。
</p>
</td> </tr> </table>
```

在这种情况下，HTML 块以空行结束 - \*\*你好\*\*文本保持逐字逐句——常规解析恢复，并加一个段落，强调世界以及内联和块 HTML 跟随。

所有类型的HTML 块除了类型 7 可以中断段落之外。类型 7 的块不得中断段落。（此限制旨在防止将包装段落内的长标签不必要地解释为起始 HTML 块。）

以下是一些简单的例子。以下是一些类型 6 的基本 HTML 块：

#### 示例 149

```
<表格>
· <tr>
· . <td>
..... 你好
..... <td> 复制代码
· </tr>
</table>

好的。
```

```
<表格>
· <tr>
· . <td>
..... 你好
..... <td> 复制代码
· </tr>
</table>
<p>好的。</p>
```

#### 示例 150

```
<div>
· * 你好*
..... <foo><a>
..... <div>
```

```
<div>
· * 你好*
..... <foo><a>
..... <div>
```

块也可以以结束标签开始：

#### 示例 151

```
</div>
* foo *
```

```
</div>
* foo *
```

这里我们有两个 HTML 块，它们之间有一个 Markdown 段落：

#### 示例 152

```
<DIV ·类= “foo” >
* 降价*
</DIV>

<DIV ·类= “foo” >
<p><em>Markdown</em></p>
</DIV>
```

第一行的标签可以是部分的，只要在有空格的地方拆分即可：

#### 示例 153

```
<div ·id="foo"
· 类= “酒吧” >
</div>
```

```
<div ·id="foo"
· 类= “酒吧” >
</div>
```

#### 示例 154

```
<div ·id="foo" ·类= “栏
· 巴兹">
</div>
```

```
<div ·id="foo" ·类= “栏
· 巴兹">
</div>
```

打开的标签不需要关闭：

#### 示例 155

```
<div>
* foo *
* 酒吧*
```

部分标签甚至不需要完成（垃圾进，垃圾出）：

#### 示例 156

```
<div ·id="foo"
* 你好*
```

#### 示例 157

```
<div ·班级
富
```

初始标签甚至不需要是有效标签，只要它以如下形式开头即可：

#### 示例 158

```
<div ·* ???-&&&-<---
* foo *
```

在类型 6 的块中，初始标签不需要单独占一行：

#### 示例 159

```
<div><a ·href="bar">*foo*</a></div>
```

```
<div><a ·href="bar">*foo*</a></div>
```

#### 示例 160

```
<表格><tr><td>
富
</td></tr></table>
```

```
<表格><tr><td>
富
</td></tr></table>
```

直到下一个空白行或文档末尾的所有内容都包含在 HTML 块中。因此，在下面的例子中，看起来像 Markdown 代码块的内容实际上是 HTML 块的一部分，它会一直持续到空白行或文档末尾：

#### 示例 161

```
<div></div>
```
丙
整数十⇒ 33;
```

```

要开始HTML 块 带有标签不在 (6) 中的块级标签列表中，必须将标签单独放在第一行（并且必须是完整的）：

#### 示例 162

```
<—href="foo">
* 酒吧*
</—>
```

```
<—href="foo">
* 酒吧*
</—>
```

在类型 7 区块中，标签名称 可以是任何东西：

#### 示例 163

```
<警告>
* 酒吧*
</警告>
```

```
<警告>
* 酒吧*
</警告>
```

#### 示例 164

```
<—类= “foo” >  
* 酒吧*  
</—>
```

```
<—类= “foo” >  
* 酒吧*  
</—>
```

## 示例 165

```
</插件>  
* 酒吧*
```

这些规则旨在让我们能够使用可以用作块级或内联级标签的标签。<删除>标签就是一个很好的例子。我们可以用<包围内容删除>标签有三种不同的方式。在本例中，我们得到一个原始 HTML 块，因为<删除>标签单独占一行：

## 示例 166

```
<删除>  
* foo *  
</del>
```

```
<删除>  
* foo *  
</del>
```

在这种情况下，我们得到一个仅包含<的原始 HTML 块删除>标签（因为它以以下空行结尾）。因此内容被解释为 CommonMark：

## 示例 167

```
<删除>  
* foo *  
</del>
```

```
<删除>  
<p><em>foo</em></p>  
</del>
```

最后，在这种情况下，<删除>标签被解释为[原始 HTML](#) 里面CommonMark 段落。（由于标签本身不单独成行，因此我们得到的是内联 HTML，而不是[HTML 块](#)。

## 示例 168

```
<del>*foo*</del>
```

```
<p><del><em>foo</em></del></p>
```

用于包含文字内容的 HTML 标签（pre, 脚本, 样式, 文本区域),注释、处理指令和声明的处理方式略有不同。这些块不是以第一个空白行结束，而是以包含相应结束标记的第一行结束。因此，这些块可以包含空白行：

预标签（类型 1）：

## 示例 169

```
<前 · 语言= “haskell” ><code>导  
入 · 文本.HTML.TagSoup
```

```
<前 · 语言= “haskell” ><code>导  
入 · 文本.HTML.TagSoup
```

```
主要的 · 输入输出 · ()  
主要的 打印 · $ 解析标签 · 标签  
</代码> </ pre>  
好的
```

```
主要的 · 输入输出 · ()  
主要的 打印 · $ 解析标签 · 标签  
</代码> </ pre>  
<p>好的</p>
```

脚本标签（类型 1）：

## 示例 170

```
<脚本 · 类型= “文本/javascript” >//  
JavaScript · 例子
```

```
<脚本 · 类型= “文本/javascript” >//  
JavaScript · 例子
```

```
document.getElementById("demo").innerHTML · = · “你好 · document.getElementById("demo").innerHTML · = · “你好 ·  
JavaScript! “;  
</script>  
好的
```

```
document.getElementById("demo").innerHTML · = · “你好 · document.getElementById("demo").innerHTML · = · “你好 ·  
JavaScript! “;  
</script>  
<p>好的</p>
```

textarea 标签（类型 1）：

## 示例 171

<文本区域>

<文本区域>

\* foo \*

\* foo \*

\_酒吧\_

\_酒吧\_

</文本区域>

</文本区域>

样式标签（类型 1）：

#### 示例 172

<样式

• 类型=“文本/css” >

h1 {颜色: 红色; }

页{颜色: 蓝色; }

</style>

好的

<样式

• 类型=“文本/css” >

h1 {颜色: 红色; }

页{颜色: 蓝色; }

</style>

<p>好的</p>

如果没有匹配的结束标记，则块将在文档末尾结束（或封闭的[块引用](#)或者[列表项](#)）：

#### 示例 173

<样式

• 类型=“文本/css” >

富

<样式

• 类型=“文本/css” >

富

#### 示例 174

> <div>

> 富

酒吧

<blockquote>

<div>

富

</blockquote>

<p>酒吧</p>

#### 示例 175

- <div>

- 富

<ul>

<li>

<div>

</li>

<li>foo</li>

</ul>

结束标记可以与开始标记出现在同一行：

#### 示例 176

<style>p{颜色: 红色;}</style>

\* foo \*

<style>p{color:red;}</style>

<p><em>foo</em></p>

#### 示例 177

<!-- \* 富 \* --> \* 酒吧\*

\*巴兹\*

<!-- \* 富 \* --> \* 酒吧\*

<p><em>巴兹</em></p>

请注意，结束标记后最后一行的所有内容都将包含在[HTML 块](#)：

#### 示例 178

<脚本>

富

</script>1. \* 酒吧\*

<脚本>

富

</script>1. \* 酒吧\*

评论（类型 2）：

#### 示例 179

<!-- · 福

<!-- · 福

酒吧  
· 巴兹 · ->  
好的

酒吧  
· 巴兹 · ->  
<p>好的</p>

处理指令 (类型3) :

#### 示例 180

```
<?php  
    · 回声 · >';  
>  
好的
```

```
<?php  
    · 回声 · >';  
>  
<p>好的</p>
```

声明 (类型 4) :

#### 示例 181

```
<!文档类型 · HTML>  
<!文档类型 · HTML>
```

CDATA (类型 5) :

#### 示例 182

```
<![CDATA[  
功能 ·   matchwo(a,b)  
{  
    · 如果(←`b && →`0) 然后 {  
    . . . 返回 · 1;  
  
    · ] · 北京经济学院 ·  
    . . . 返回 · 0;  
    . } .  
}  
]]>  
好的
```

```
<![CDATA[  
功能 ·   matchwo(a,b)  
{  
    · 如果(←`b && →`0) 然后 {  
    . . . 返回 · 1;  
  
    · ] · 北京经济学院 ·  
    . . . 返回 · 0;  
    . } .  
}  
]]>  
<p>好的</p>
```

开始标签前面最多可以有三个缩进空格，但不能有四个：

#### 示例 183

```
· <!-- · 哟 · ->  
· . <!-- · 哟 · ->  
· . . <!-- · 哟 · ->  
  
· <!-- · 哟 · ->  
<pre><code><!-- · 富 · ->  
</代码> </ pre>
```

#### 示例 184

```
· <div>  
· . <div>  
. . . <div>  
  
· <div>  
<pre><code><div>  
</代码> </ pre>
```

类型 1-6 的 HTML 块可以中断段落，并且前面不需要有空行。

#### 示例 185

```
福  
<div>  
酒吧  
</div>  
  
<p>食物</p>  
<div>  
酒吧  
</div>
```

但是，除了在文档末尾以及 1-5 类型的块之外，后面还需要一个空行，多于：

#### 示例 186

```
<div>  
酒吧  
</div>  
* foo *
```

```
<div>  
酒吧  
</div>  
* foo *
```

类型 7 的 HTML 块不能中断段落：

#### [示例 187](#)

```
福  
<—href="酒吧">  
巴兹
```

```
<p>Foo  
<—href="酒吧">  
</p>
```

此规则与 John Gruber 最初的 Markdown 语法规范不同，其内容如下：

唯一的限制是块级 HTML 元素 — 例如 `<div>`, `<table>`, `<pre>`, `<p>`, 等——必须用空行与周围内容分开，并且块的开始和结束标签不应该用空格或制表符缩进。

在某些方面，格鲁伯的规则比这里给出的规则更严格：

- 它要求 HTML 块前面有一个空行。它不允许缩进开始标记。
- 
- 它需要匹配的结束标签，但也不允许缩进。

大多数 Markdown 实现（包括 Gruber 自己的一些实现）并不遵守所有这些限制。

然而，Gruber 的规则在某个方面比这里给出的规则更宽松，因为它允许在 HTML 块内出现空行。这里不允许空行有两个原因。首先，它消除了解析平衡标签的需要，这种解析很昂贵，并且如果找不到匹配的结束标签，可能需要从文档末尾回溯。其次，它提供了一种非常简单灵活的方法将 Markdown 内容包含在 HTML 标签内：只需使用空行将 Markdown 与 HTML 分开即可：

比较：

#### [示例 188](#)

```
<div>  
*强调* · 文本。  
</div>
```

```
<div>  
<p><em>强调</em> · 文本。</p>  
</div>
```

#### [示例 189](#)

```
<div>  
*强调* · 文本。  
</div>
```

```
<div>  
*强调* · 文本。  
</div>
```

一些 Markdown 实现采用了一种惯例，即如果打开的标签具有以下属性，则将标签内的内容解释为文本降级=1。上面给出的规则看起来是一种更简单、更优雅的方式来实现相同的表达能力，而且解析起来也简单得多。

主要的潜在缺点是，不再能够 100% 可靠地将 HTML 块粘贴到 Markdown 文档中。不过，大多数情况下这样做没问题，因为 HTML 中的空行后面通常跟着 HTML 块标记。例如：

#### [示例 190](#)

```
<表格>  
<tr>  
<td>  
你好  
<td> 复制代码  
</tr>  
</table>
```

```
<表格>  
<tr>  
<td>  
你好  
<td> 复制代码  
</tr>  
</table>
```

然而，如果内部标签缩进，就会出现问题和用空格分隔，因为它们将被解释为缩进的代码块：

### 示例 191

```
<表格>
  <tr>
    <td>你好</td>
  </tr>
</table>
```

幸运的是，空行通常不是必需的，可以删除。例外情况是 `<里面上一篇>` 标签，但如所述 [多于](#)，以 `<` 开头的原始 HTML 块上一篇能包含空行。

## 4.7 链接引用定义

一个 [链接引用定义](#) 包括一个 [链接标签](#)，前面最多可加三个缩进空格，后跟一个冒号 (:)，后面可加空格或制表符（包括最多一个 [行结束符](#)），一个 [链接目标](#)、可选空格或制表符（包括最多一个 [行结束符](#)）以及可选的 [链接标题](#)，如果存在，必须与 [链接目标](#) 空格或制表符。不能再出现其他字符。

一个 [链接引用定义](#) 不对应于文档的结构元素。相反，它定义了一个标签，可用于 [参考链接](#) 和引用风格 [图片](#) 文档的其他地方。[链接参考定义](#) 可以位于使用它们的链接之前或之后。

### 示例 192

```
[foo]: ./网址 ·“标题”<p><a href="/网址" · title="title">foo</a></p>
```

```
[foo]
```

### 示例 193

```
. . [foo]: ·
. . . ./网址 ·
. . . . '这 : 标题' ·<p><a href="/网址" · 标题=" · 标题">foo</a></p>
```

```
[foo]
```

### 示例 194

```
[Foo*bar\]]:my_(url) · '标题' · (和 · 括号) ·<p><a href="我的_(url)" · title="标题" · (和 ·
[括号) "> Foo * bar]</a></p>
```

```
[Foo*bar\]]
```

### 示例 195

```
[富 · 酒吧]:
<我的网址>
'标题'<p><a href="我的网址" · 标题= “标题” > Foo · </a> 酒吧
</p>
```

```
[富 · 酒吧]
```

标题可以跨越多行：

### 示例 196

```
[foo]: ./网址 ·‘
标题
第 1 行
第2行
’<p><a href="/网址" · 标题="标题
第 1 行
第2行
">foo</a></p>
```

```
[foo]
```

然而，它可能不包含 [空行](#)：

## 示例 197

```
[foo]: ·/网址'标题  
和 · 空白的线'  
<p>[foo]: ·/网址'标题</p>  
<p>与 · 空白的线'</p>  
<p>[foo]</p>
```

```
[foo]
```

可以省略标题：

## 示例 198

```
[foo]:  
/网址  
<p><a href="/url">foo</a></p>  
[foo]
```

链接目的地不能被省略：

## 示例 199

```
[foo]:  
[foo]  
<p>[foo]:</p>  
<p>[foo]</p>
```

但是，可以使用尖括号指定空的链接目标：

## 示例 200

```
[foo]: ·<>  
[foo]  
<p><a href="">foo</a></p>
```

标题必须与链接目标用空格或制表符分隔：

## 示例 201

```
[foo]: ·<bar> (巴兹)  
[foo]  
<p>[foo]: ·<bar> (巴兹) </p>  
<p>[foo]</p>
```

标题和目标都可以包含反斜杠转义符和文字反斜杠：

## 示例 202

```
[foo]: ·/url\bar\*baz · "foo\" bar\baz"  
[foo]  
<p><a href="/url%5Cbar%5Cbaz" ·  
title="foo"bar\baz">foo</a></p>
```

链接可以位于其对应的定义之前：

## 示例 203

```
[foo]  
[foo]: ·网址  
<p><a href="url">foo</a></p>
```

如果有多个匹配的定义，则第一个优先：

## 示例 204

```
[foo]  
[foo]: ·第一的  
[foo]: ·第二  
<p><a href="first">foo</a></p>
```

正如在[链接](#)，标签匹配不区分大小写（参见[火柴](#)）。

## 示例 205

[FOO]: ./网址

<p><a href="/url">Foo</a></p>

[福]

## 示例 206

[ΑΓΩ]: . /φου

<p><a href="/%CF%86%CE%BF%CF%85">αγω</a></p>

[αγω]

不管某件事链接引用定义 与其定义的链接引用是否在文档中使用无关。因此，例如，以下文档仅包含链接引用定义，不包含可见内容：

## 示例 207

[foo]: .-/网址

这是另一个：

## 示例 208

[  
富  
]: /网址  
酒吧

<p>酒吧</p>

这不是链接引用定义，因为标题后面除了空格或制表符之外还有其他字符：

## 示例 209

[foo]: .-/网址 “标题” ·好的

<p>[foo]: .-/网址 ·“标题” ·  
好的</p>

这是一个链接引用定义，但是没有标题：

## 示例 210

[foo]: .-/网址  
“标题” ·好的

<p> “标题” ·  
好的</p>

这不是一个链接引用定义，因为它缩进四个空格：

## 示例 211

. . . [foo]: .-/网址 ·“标题”

<pre><code>[foo]: .-/网址 ·“标题”  
</code></pre>  
<p>[foo]</p>

[foo]

这不是链接引用定义，因为它出现在代码块内：

## 示例 212

```  
[foo]: .-/网址

<pre><code>[foo]: .-/网址  
</code></pre>  
<p>[foo]</p>

[foo]

一个链接引用定义 不能打断段落。

## 示例 213

福  
[酒吧]: ./巴兹  
[酒吧]

<p>Foo  
[酒吧]: ./巴兹</p>  
<p>[酒吧]</p>

但是，它可以直接跟在其他块元素之后，例如标题和主题分隔符，并且它后面不需要有空行。

## 示例 214

```
# {福}
[foo]: ./网址
> 酒吧
          <h1><a href="/url">Foo</a></h1>
          <blockquote>
            <p>酒吧</p>
          </blockquote>
```

## 示例 215

```
[foo]: ./网址
酒吧
===
[foo]
```

```
<h1>酒吧</h1>
<p><a href="/url">foo</a></p>
```

## 示例 216

```
[foo]: ./网址
===
[foo]
```

```
<p>===
<—href="/url">foo</a></p>
```

一些链接参考定义可以一个接一个地出现，中间不插入空行。

## 示例 217

```
[foo]: ./foo-url • "foo"
[酒吧]: ./bar-url
  • "酒吧"
[巴兹]: ./baz-url
```

```
<p><a href="/foo-url" • title="foo">foo</a>,
<—href="/bar-url" • title="bar">酒吧</a>,
<—href="/baz-url">巴兹</a></p>
```

```
[foo],
[酒吧],
[巴兹]
```

链接参考定义可以出现在块容器内，如列表和块引用。它们影响整个文档，而不仅仅是定义它们的容器：

## 示例 218

```
[foo]
> {foo}: ./网址
```

```
<p><a href="/url">foo</a></p>
          <blockquote>
            </blockquote>
```

## 4.8 段落

不能被解释为其他类型块的非空行序列形成段落。段落的内容是将段落的原始内容解析为内联的结果。段落的原始内容是通过连接行并删除首尾空格或制表符形成的。

包含两段内容的简单示例：

## 示例 219

```
AAA
bbb
```

```
<p>啊</p>
<p>bbb</p>
```

段落可以包含多行，但不能有空行：

## 示例 220

```
AAA
bbb
加拿大发售委员会
ddd
```

```
<p>aaa
</p>
<p>ccc</p>
ddd</p>
```

段落之间的多个空行没有任何效果：

## 示例 221

```
AAA          <p>啊</p>
             <p>bbb</p>
```

bbb

跳过前导空格或制表符：

## 示例 222

```
• AAA          <p>aaa
bbb           </p>
```

第一行之后的行可以缩进任意量，因为缩进的代码块不能打断段落。

## 示例 223

```
AAA          <p>aaa
.....bbb    bbb
.....加拿大合作委员会.....</p>.....
```

但是，第一行前面最多可以有三个缩进。四个缩进太多了：

## 示例 224

```
.. AAA          <p>aaa
bbb           </p>
```

## 示例 225

```
... AAA          <pre><code>aaa
bbb           </code></pre>
             <p>bbb</p>
```

在内联解析之前，最后的空格或制表符会被删除，因此以两个或更多空格结尾的段落不会以硬换行：

## 示例 226

```
AAA.....      <p>啊啊啊啊啊<br />
bbb.....      </p>
```

## 4.9 空行

空行 块级元素之间的区别被忽略，除了它们在确定一个[列表是紧的](#)或者[松动的](#)。

文档开始和结束处的空行也会被忽略。

## 示例 227

```
...
AAA          <p>啊</p>
             <h1>啊</h1>
...
# AAA
...
```

## 5 容器块

一个[容器块](#)是一个以其他块为内容的块。容器块有两种基本类型：[块引用](#)和[列表项](#)。[列表](#)是元容器[列表项](#)。

我们递归地定义容器块的语法。定义的一般形式为：

如果 X 是一系列块，那么以某种方式转换 X 的结果是一个类型为 Y 的容器，其中包含这些块作为其内容。

因此，我们通过解释如何将其视为区块引用或列表项来解释什么算作区块引用或列表项生成从其内容中。这应该足以定义语法，尽管它没有给出一个秘诀解析这些构造。（下面标题为[解析策略](#)）。

## 5.1 区块引用

一个[块引用标记](#)，可选择在前面加上最多三个缩进空格，由 (a) 字符 > 和后面的缩进空格，或 (b) 单个字符 > 后面不跟缩进空格组成。

以下规则定义[区块引用](#)：

1. **基本情况。** 如果一串行负责构成一个块序列贝斯，然后添加一个[块引用标记](#) 到每行的开头负责是[区块引用](#) 包含贝斯。

2. **懒惰。** 如果一串行负责构成[区块引用](#) 包含内容贝斯，然后删除初始[块引用标记](#) 来自一行或多行，其中在行尾后有一个除空格或制表符之外的下一个字符[块引用标记](#) 是[段落延续文本](#) 是带有贝斯作为其内容。[段落延续文本](#) 是将被解析为段落内容的一部分但不会出现在段落开头的文本。

3. **连续性。** 文档不能包含两个[区块引用](#) 除非有[空行](#) 他们之间。

其他任何事情都不算[区块引用](#)。

这是一个简单的例子：

### 示例 228

```
> # 福                                <blockquote>
> 酒吧                                <h1>Foo</h1>
> 巴兹                                <p>酒吧
                                         </p>
                                         </blockquote>
```

> 字符后的空格或制表符可以省略：

### 示例 229

```
> #福                                <blockquote>
>酒吧                                <h1>Foo</h1>
> 巴兹                                <p>酒吧
                                         </p>
                                         </blockquote>
```

> 字符前面最多可以有三个缩进空格：

### 示例 230

```
. .>. # 哟                            <blockquote>
. .> 酒吧                                <h1>Foo</h1>
> 巴兹                                <p>酒吧
                                         </p>
                                         </blockquote>
```

四个空格的缩进太多了：

### 示例 231

```
. . .>. # 哟                         <pre><code>> . # 福
. . .> 酒吧                                > . 酒吧
. . .> 巴兹                                > . 巴兹
                                         </code></pre>
```

惰性子句允许我们省略前面的[段落延续文本](#)：

### 示例 232

```
> # 福  
> 酒吧  
巴兹
```

```
<blockquote>  
<h1>Foo</h1>  
<p>酒吧  
</p>  
</blockquote>
```

块引用可以包含一些惰性延续行和一些非惰性延续行：

### [示例 233](#)

```
> 酒吧  
巴兹  
> 富
```

```
<blockquote>  
<p>酒吧  
巴兹  
</p>  
</blockquote>
```

惰性仅适用于那些本来是段落延续的行，如果它们前面加上[区块引用标记](#)。例如，第二行中的 > 不能省略

```
> foo  
>---
```

意思不变：

### [示例 234](#)

```
> 富  
---  
-> 酒吧
```

```
<blockquote>  
<p>foo</p>  
</blockquote>  
<小时>。
```

类似地，如果我们省略第二行中的 >

```
>-foo  
>-酒吧
```

然后块引用在第一行之后结束：

### [示例 235](#)

```
> - 富  
- 酒吧
```

```
<blockquote>  
<ul>  
<li>foo</li>  
</ul>  
</blockquote>  
<ul>  
<li>酒吧</li>  
</ul>
```

出于同样的原因，我们不能省略缩进或围栏代码块后续行前面的 >：

### [示例 236](#)

```
> . . . 富  
. . . 酒吧
```

```
<blockquote>  
<pre><code>foo  
</code></pre>  
</blockquote>  
<pre><code>条形码  
</code></pre>
```

### [示例 237](#)

```
> . ``  
富  
. . .
```

```
<blockquote>  
<pre><code></code></pre>  
</blockquote>  
<p>foo</p>  
<pre><code></code></pre>
```

请注意，在以下情况下，我们有一个[惰性延续行](#)：

## 示例 238

```
> 富
. . . - b应收账款
          <blockquote>
          <p>foo
          - </p>
          </blockquote>
```

要了解原因，请注意

```
> foo
>     - 酒吧
```

这 - 酒吧缩进太多，无法开始列表，也不能是缩进的代码块（因为缩进的代码块不能中断段落），所以[段落延续文本](#)。

块引用可以为空：

## 示例 239

```
>
          <blockquote>
          </blockquote>
```

## 示例 240

```
>
> .
> .
          <blockquote>
          </blockquote>
```

块引用可以有开头或结尾的空白行：

## 示例 241

```
>
> 富
> .
          <blockquote>
          <p>foo</p>
          </blockquote>
```

块引用之间总是有一个空行：

## 示例 242

```
> 富
> 酒吧
          <blockquote>
          <p>foo</p>
          </blockquote>
          <blockquote>
          <p>酒吧</p>
          </blockquote>
```

(大多数当前的 Markdown 实现，包括 John Gruber 的原始Markdown.pl，将此示例解析为包含两个段落的单个块引用。但似乎最好让作者决定是否需要两个块引用或一个块引用。)

连续性意味着如果我们将这些块引用放在一起，我们会得到一个块引用：

## 示例 243

```
> 富
> 酒吧
          <blockquote>
          <p>foo
          </p>
          </blockquote>
```

要获得包含两段的块引用，请使用：

## 示例 244

```
> 富
>
> 酒吧
          <blockquote>
          <p>foo</p>
          <p>酒吧</p>
          </blockquote>
```

块引用可以打断段落：

#### 示例 245

富  
> 酒吧

```
<p>foo</p>
<blockquote>
<p>酒吧</p>
</blockquote>
```

一般来说，块引用之前或之后不需要空行：

#### 示例 246

> AAA  
\*\*\*  
> bbb

```
<blockquote>
<p>啊</p>
</blockquote>
<小时>
<blockquote>
<p>bbb</p>
</blockquote>
```

但是，由于懒惰，块引用和后续段落之间需要一个空行：

#### 示例 247

> 酒吧  
巴兹

```
<blockquote>
<p>酒吧
</p>
</blockquote>
```

#### 示例 248

> 酒吧  
巴兹

```
<blockquote>
<p>酒吧</p>
</blockquote>
<p>巴兹</p>
```

#### 示例 249

> 酒吧  
>  
巴兹

```
<blockquote>
<p>酒吧</p>
</blockquote>
<p>巴兹</p>
```

惰性规则的结果是，在嵌套块引用的连续行上可以省略任意数量的初始 >：

#### 示例 250

> > > 富  
酒吧

```
<blockquote>
<blockquote>
<blockquote>
<p>foo
</p>
</blockquote>
</blockquote>
</blockquote>
```

#### 示例 251

>>> 富  
> 酒吧  
>> 巴兹

```
<blockquote>
<blockquote>
<blockquote>
<p>foo
酒吧
</p>
</blockquote>
</blockquote>
</blockquote>
```

在块引用中包含缩进的代码块时，请记住[块引用标记](#)包括 > 和后面的缩进空格。因此五个空格在 > 之后需要：

## 示例 252

> . . 代码  
> . . 不是丙颂

```
<blockquote>  
<pre><code>代码  
</code> </pre>  
</blockquote>  
<blockquote>  
<p>不是 · 代码</p>  
</blockquote>
```

## 5.2 列出项目

一个列表标记是项目符号列表标记或有序列表标记。

一个项目符号列表标记是 -、+ 或 \* 字符。

一个有序列表标记是 1-9 个阿拉伯数字的序列 (0-9) 后面跟着一个 . 字符或 ) 字符。(长度限制的原因是，在某些浏览器中，当数字达到 10 位时，我们就会开始看到整数溢出。)

以下规则定义列表项：

1. **基本情况。**如果一系列行负责构成一个块序列贝斯以空格或制表符以外的字符开头，并且米是宽度的列表标记，则接下来是 1 ≤ 否 ≤ 4 个空格的缩进，然后添加前置结果米并将以下空格添加到第一行负责，并缩进后续行负责经过无线+空格，是一个列表项贝斯作为其内容。列表项的类型（项目符号或有序）由其列表标记的类型决定。如果列表项是有序的，则还会根据有序列表标记为其分配一个起始编号。

例外情况：

1. 当列表打断一个段落——也就是说，当一个段落开始于一个原本算作段落延续文本 —然后 (a) 行负责不能以空行开头，并且 (b) 如果列表项是有序的，则起始编号必须为 1。
2. 如果任何一条线是专题休息 那么该行就不是列表项。

例如，让负责成为线条

## 示例 253

一段落  
和 · 二 · 線。  
. . 缩进 · 代码  
> —堵塞 · 引用。

```
<p>—一个 段落  
和 · 二 · 行。</p>  
<pre><code>缩进 · 代码  
</code> </pre>  
<blockquote>  
<p>—堵塞 · 引文。</p>  
</blockquote>
```

并让米成为标记1.，和否=2. 规则 1 规定，以下是一个有序列表项，起始编号为 1，内容与负责：

## 示例 254

1. · 一段落  
. . 和 · 二 · 線。  
. . . . 缩进 · 代码  
. . . . > 乙·锁 · 引用。

```
<ol>  
<li>  
<p>—一个 段落  
和 · 二 · 行。</p>  
<pre><code>缩进 · 代码  
</code> </pre>  
<blockquote>  
<p>—堵塞 · 引文。</p>  
</blockquote>  
</li>  
</ol>
```

需要注意的最重要的一点是，列表标记后的文本位置决定了列表项中后续块需要缩进多少。如果列表标记占用两个缩进空格，并且列表标记与下一个字符（空格或制表符除外）之间有三个空格，则块必须缩进五个空格才能落入列表项下。

以下一些示例显示了内容必须缩进多远才能放置在列表项下：

### 示例 255

```
<ul>
<li>一个</li>
</ul>
<p>二</p>
```

### 示例 256

```
<ul>
<li>
<p>一</p>
<p>二</p>
</li>
</ul>
```

### 示例 257

```
<ul>  
<li>一个</li>  
</ul>  
<pre><代码> ·  
</代码></pre>
```

## 示例 258

```
<ul>
<li>
<p>一</p>
<p>二</p>
</li>
</ul>
```

很容易从列的角度来思考这个问题：连续块必须至少缩进到列表标记后第一个字符（空格或制表符除外）的列。然而，这并不完全正确。列表标记后的缩进空间决定了需要多少相对缩进。缩进到达哪一列将取决于列表项如何嵌入到其他结构中，如下例所示：

### 示例 259

```
 . . > > 1. + --  
>>  
>> . . . __
```

<blockquote>  
<blockquote>  
<ol>  
<li>  
<p>一</p>  
<p>二</p>  
</li>  
</ol>  
</blockquote>  
</blockquote>

这里二与列表标记位于同一列1., 但实际上包含在列表项中, 因为在最后一个包含的块引用标记后有足够的缩进。

反过来也是可以的。在下面的例子中，单词二出现在列表项初始文本的右侧很远的地方，一，但它不被视为列表项的一部分，因为它的缩进距离不够远，超出了块引用标记：

## 示例 260

```
> >- ←  
> >  
• >.. >..__  
  
<blockquote>  
<blockquote>  
<ul>  
<li>—一个</li>  
</ul>  
<p>__</p>  
</blockquote>  
</blockquote>
```

请注意，列表标记和任何后续内容之间至少需要一个空格或制表符，因此这些不是列表项：

## 示例 261

- —  
2. —

列表项可能包含由多个空自行分隔的块。

## 示例 262

```
- 富
  • 酒吧
```

<ul>  
 <li><p>foo</p>  
 <li><p>酒吧</p>  
</li>  
</ul>

列表项可以包含任何类型的块：

### 示例 263

|             |                                   |
|-------------|-----------------------------------|
| 1. • 富      | <ol>                              |
| . . . . `   | <里>                               |
| . . . 酒吧    | <p>foo</p>                        |
| . . . . `   | <pre><code>条形码</code></pre>       |
| . . . 巴茲    | <p>巴茲</p>                         |
| . . . > bam | <blockquote><p>碑</p></blockquote> |
|             | </li>                             |
|             | </ol>                             |

包含缩进代码块的列表项将逐字保留代码块内的空行。

### 示例 264

- 福
  - .... 酒吧
  - .... 巴兹

请注意，有序列表的起始数字必须小于或等于 9 位

### 示例 265

123456789。 ·好的  
<ol ·开始="123456789"  
  <li>好的</li>  
  </ol>

### 示例 266

1234567890。 ·不是好的 <p>1234567890。 ·不是好的</p>

起始数字可能以 0 开头：

### 示例 267

0。好的  
<ol><li>好的</li></ol>

示例 268

003. ·好的

```
<ol ·开始= “3” >  
<li>好的</li>  
</ol>
```

起始数字不能为负数：

#### [示例 269](#)

- 1. ·不是好的

```
<p>-1. ·不是好的</p>
```

2. **以缩进代码开头的项目。**如果一系列行负责构成一个块序列，以缩进的代码块开始，并且它是宽度的列表标记后面跟着一个缩进空格，然后是前面添加的结果，并将以下空格添加到第一行负责，并缩进后续行负责经过白 + 1 空格，是一个列表项。作为其内容。如果某行是空的，则无需缩进。列表项的类型（项目符号或有序）由其列表标记的类型决定。如果列表项是有序的，则还会根据有序列表标记为其分配一个起始编号。

缩进的代码块前面必须在列表项中包含文本的区域边缘之外留出四个空格的缩进。在以下情况下，缩进量为 6 个空格：

#### [示例 270](#)

- 富

```
<ul>  
<li>  
<p>foo</p>  
<pre><code>条形码  
</code></pre>  
</li>  
</ul>
```

在这种情况下，它是 11 个空格：

#### [示例 271](#)

· 10. · f 哦

```
<ol ·开始= “10” >  
<li>  
<p>foo</p>  
<pre><code>条形码  
</code></pre>  
</li>  
</ol>
```

如果第一的列表项中的 block 是缩进的代码块，那么根据规则 2，内容前面必须有一列表标记后的缩进空格：

#### [示例 272](#)

... 缩进 · 代码

```
<pre><code>缩进 · 代码  
</code></pre>  
<p>段落</p>  
<pre><code>更多 · 代码  
</code></pre>
```

段落

... 更多的代码

#### [示例 273](#)

1. .... 缩进 · 代码

```
<ol>  
<li>  
<pre><code>缩进 · 代码  
</code></pre>  
<p>段落</p>  
<pre><code>更多 · 代码  
</code></pre>  
</li>  
</ol>
```

.. 段落

.... 更多的丙倾

请注意，缩进的额外空格将被解释 为代码块内的空格：

#### [示例 274](#)

```
1.....缩进 · 代码
. . 段落
. . . . 更多的丙颂
<ol>
<里>
<pre><代码> · 缩进 · 代码
</代码> </ pre>
<p>段落</p>
<pre><code>更多 · 代码
</代码> </ pre>
</li>
</ol>
```

请注意，规则 #1 和 #2 仅适用于两种情况：(a) 列表项中包含的行以空格或制表符以外的字符开头的情况，以及 (b) 以缩进的代码块开头的情况。在以下情况下，第一个块以三个缩进空格开头，规则不允许我们通过缩进整个内容并添加列表标记来形成列表项：

### [示例 275](#)

```
. . 富
<p>foo</p>
<p>酒吧</p>
酒吧
```

### [示例 276](#)

```
- . . 富
. 酒吧
<ul>
<li>foo</li>
</ul>
<p>酒吧</p>
```

这不是一个重要的限制，因为当一个块前面有最多三个缩进空格时，可以始终删除缩进而不会改变解释，从而允许应用规则 1。因此，在上述情况下：

### [示例 277](#)

```
- . 富
. . 酒吧
<ul>
<里>
<p>foo</p>
<p>酒吧</p>
</li>
</ul>
```

**3.项目以空白行开始。**如果一系列行负责从单一开始**空行**构成一个（可能为空的）块序列贝斯，和米是宽度的列表标记西，然后添加结果米到第一行负责以及前面的后续行负责经过白 + 1缩进空格，是列表项贝斯作为其内容。如果某行是空的，则无需缩进。列表项的类型（项目符号或有序）由其列表标记的类型决定。如果列表项是有序的，则还会根据有序列表标记为其分配一个起始编号。

以下是一些以空白行开头但不为空的列表项：

### [示例 278](#)

```
- . 富
- . . 酒吧
. . . 巴兹
<ul>
<li>foo</li>
<里>
<pre><code>条形码
</代码> </ pre>
</li>
<里>
<pre><code>baz
</代码> </ pre>
</li>
</ul>
```

当列表项以空行开始时，列表标记后的空格数不会改变所需的缩进：

### [示例 279](#)

```
<ul>
<li>foo</li>
</ul>
```

列表项最多可以以一个空行开头。在下面的示例中，富不是列表项的一部分：

#### 示例 280

```
-  
· 富  
  
<ul>  
<li></li>  
</ul>  
<p>foo</p>
```

这是一个空的项目符号列表项：

#### 示例 281

```
- 富  
- 酒吧  
  
<ul>  
<li>foo</li>  
<li></li>  
<li>酒吧</li>  
</ul>
```

后面是否有空格或制表符并不重要[列表标记](#)：

#### 示例 282

```
- 富  
- . . .  
- 酒吧  
  
<ul>  
<li>foo</li>  
<li></li>  
<li>酒吧</li>  
</ul>
```

这是一个空的有序列表项：

#### 示例 283

```
1. ·富  
2.  
3. ·酒吧  
  
<ol>  
<li>foo</li>  
<li></li>  
<li>酒吧</li>  
</ol>
```

列表可以以空列表项开始或结束：

#### 示例 284

```
*  
  
<ul>  
<li></li>  
</ul>
```

但是，空列表项不能中断段落：

#### 示例 285

```
富  
*  
  
富  
1.  
  
<p>foo  
*</p>  
<p>foo  
1.</p>
```

**4. 缩进。**如果一系列行负责根据规则 #1、#2 或 #3 构成列表项，则在每一行前面加上负责最多缩进三个空格（每行相同），也构成具有相同内容和属性的列表项。如果一行是空的，则无需缩进。

缩进一个空格：

#### 示例 286

1. · 一段落  
... 和 · 二 · 線。  
..... 缩进 · 代码  
....> 乙 · 锁 · 引用。

```
<ol>
<li>
<p>一个段落
和 · 二 · 行。</p>
<pre><code>缩进 · 代码
</code></pre>
<blockquote>
<p>一个堵塞 · 引文。</p>
</blockquote>
</li>
</ol>
```

缩进两个空格：

#### 示例 287

· 1.. 磷 · 译文  
... 和 · 二 · 線。  
..... 缩进 · 代码  
....> 乙 · 锁 · 引用。

```
<ol>
<li>
<p>一个段落
和 · 二 · 行。</p>
<pre><code>缩进 · 代码
</code></pre>
<blockquote>
<p>一个堵塞 · 引文。</p>
</blockquote>
</li>
</ol>
```

缩进三个空格：

#### 示例 288

· 1. · 磷 · 译文  
... 和 · 二 · 線。  
..... 缩进 · 代码  
....> 一堵塞 · 引用。

```
<ol>
<li>
<p>一个段落
和 · 二 · 行。</p>
<pre><code>缩进 · 代码
</code></pre>
<blockquote>
<p>一个堵塞 · 引文。</p>
</blockquote>
</li>
</ol>
```

四个空格缩进形成一个代码块：

#### 示例 289

· · 1. · 磷 · 译文  
... 和 · 二 · 線。  
..... 缩进 · 代码  
....> 乙 · 锁 · 引用。

```
<pre><code>1. · 一段落
... 和 · 二 · 線。
..... 缩进 · 代码
....> 乙 · 锁 · 引用。
</code></pre>
```

5. 懒惰。如果一串行负责构成[列表项](#)包含内容贝斯，则删除一行或多行中部分或全部缩进的结果为，其中缩进后的下一个字符（空格或制表符除外）是[段落延续文本](#)是具有相同内容和属性的列表项。未缩进的行称为[惰性延续行](#)秒。

下面是一个例子[惰性延续行](#)：

#### 示例 290

• 1.. 磷·译文  
和 • 二·線。  
..... 缩进 · 代码  
.... > .乙·锁 · 引用。

```
<ol>
<li>
<p>一个段落
和 • 二·行。</p>
<pre><code>缩进 · 代码
</code></pre>
<blockquote>
<p>一个堵塞·引文。</p>
</blockquote>
</li>
</ol>
```

缩进可以部分删除：

#### 示例 291

• 1.. 磷·译文  
... 和 • 二·線。

```
<ol>
<li>
<p>一个段落
和 • 二·行。</p>
</ol>
```

这些示例展示了惰性在嵌套结构中如何发挥作用：

#### 示例 292

> 1. -> 引用  
持续 · 这里。

```
<blockquote>
<ol>
<li>
<blockquote>
<p>区块引用
持续 · 这里。</p>
</blockquote>
</li>
</ol>
</blockquote>
```

#### 示例 293

> 1. -> 引用  
> 持续 · 这里。

```
<blockquote>
<ol>
<li>
<blockquote>
<p>区块引用
持续 · 这里。</p>
</blockquote>
</li>
</ol>
</blockquote>
```

6.就这样。规则 1-5 中未算作列表项的内容均不算作列表项。

子列表的规则遵循一般规则多于。子列表必须缩进与段落相同的缩进空格数，才能将其包含在列表项中。

因此，在这种情况下我们需要缩进两个空格：

#### 示例 294

```
- 富
  - 酒吧
    - 巴兹
      - 嘘
<ul>
<li>foo
<ul>
<li>酒吧
<ul>
<li>巴兹
<ul>
<li>嘘</li>
</ul>
</li>
</ul>
</li>
</ul>
</li>
</ul>
```

一个是不够的：

#### 示例 295

```
- 富
  - 酒吧
    - 巴兹
      - 嘘
<ul>
<li>foo</li>
<li>酒吧</li>
<li>巴兹</li>
<li>嘘</li>
</ul>
```

这里我们需要四个，因为列表标记更宽：

#### 示例 296

```
10) 富
  . . . 酒吧
<ol >
<li>foo
<ul>
<li>酒吧</li>
</ul>
</li>
</ol>
```

三个还不够：

#### 示例 297

```
10) 富
  . . . 酒吧
<ol >
<li>foo</li>
</ol>
<ul>
<li>酒吧</li>
</ul>
```

列表可能是列表项中的第一个块：

#### 示例 298

```
- 富
<ul>
<li>
<ul>
<li>foo</li>
</ul>
</li>
</ul>
```

#### 示例 299

```
<ol>
<里>
<ul>
<里>
<ol *开始=“2” >
<li>foo</li>
</ol>
</li>
</ul>
</li>
</ol>
```

列表项可以包含标题：

### 示例 300

```
- # 福
- 酒吧
• • •
• 巴兹
```

```
<ul>
<里>
<h1>Foo</h1>
</li>
<里>
<h2>酒吧</h2>
</li>
</ul>
```

## 5.2.1 动机

John Gruber 的 Markdown 规范对列表项有如下说明：

1. “列表标记通常从左边距开始，但最多可缩进三个空格。列表标记后面必须跟一个或多个空格或制表符。”
2. “为了让列表看起来更美观，你可以用悬挂缩进来包装项目……但如果你不想，你不必这么做。”
3. “列表项可能由多个段落组成。列表项中的每个后续段落必须缩进 4 个空格或一个制表符。”
4. “如果缩进后续段落的每一行，看起来会很好，但是 Markdown 又会允许你偷懒。”
5. “要将块引用放在列表项中，块引用的 > 分隔符需要缩进。”
6. “要将代码块放入列表项中，代码块需要缩进两次 - 8 个空格或 2 个制表符。”

这些规则规定，列表项下的段落必须缩进四个空格（大概是从左边距开始，而不是从列表标记的开头开始，但没有说明），列表项下的代码必须缩进八个空格，而不是通常的四个空格。他们还说，块引用必须缩进，但没有说明缩进多少；然而，给出的示例有四个空格的缩进。虽然没有提到其他类型的块级内容，但可以合理地推断全部列表项下的块元素（包括其他列表）必须缩进四个空格。这一原则被称为四空间规则。

四空间规则清晰且有原则性，如果参考实现Markdown.pl如果遵循它，它可能会成为标准。然而，Markdown.pl允许段落和子列表以仅两个空格缩进开头，至少在外层。更糟糕的是，它的行为不一致：外层列表的子列表需要两个空格缩进，但此子列表的子列表需要三个空格。因此，Markdown 的不同实现开发了非常不同的规则来确定列表项下的内容，这并不奇怪。（例如，Pandoc 和 python-Markdown 坚持使用 Gruber 的语法描述和四个空格规则，而 discount、redcarpet、marked、PHP Markdown 和其他则遵循Markdown.pl的行为。）

不幸的是，鉴于实现之间的差异，没有办法为列表项提供规范，保证不会破坏任何现有文档。但是，此处给出的规范应该可以正确处理使用四空格规则或更宽容的格式的列表Markdown.pl行为，只要它们的布局方式对人类来说很自然即可。

这里的策略是让列表标记的宽度和缩进决定块在列表项下所需的缩进量，而不是使用固定的任意数字。作者可以将列表项的主体视为一个单元，该单元向右缩进到足以适合列表标记（以及任何

列表标记上的缩进）。（惰性规则 #5 允许在需要时取消连续行的缩进。）

我们声称，这条规则优于任何要求从边距固定缩进级别的规则。四空间规则很清晰，但不自然。

- foo

酒吧

-巴兹

应解析为两个列表，中间有一个段落，

```
<ul>
<li>foo</li>
</ul>
<p>酒吧</p>
<ul>
<li>巴兹</li>
</ul>
```

正如四空间规则所要求的那样，而不是单个列表，

```
<ul>
<li>
<p>foo</p>
<p>酒吧</p>
<ul>
<li>巴兹</li>
</ul>
</li>
</ul>
```

四个空格的选择是任意的。可以学习，但不太可能猜到，而且它经常让初学者犯错。

采用两空格规则会有帮助吗？问题是，这样的规则与允许初始列表标记最多缩进三个空格的规则一起，允许缩进的文本少于要包含在列表项中的原始列表标记。例如，Markdown.pl解析

- —

—

作为单个列表项，二延续段落：

```
<ul>
<li>
<p>一个</p>
<p>二</p>
</li>
</ul>
```

同样地

```
>-—
>
> 二
```

作为

```
<blockquote>
<ul>
<li>
<p>一个</p>
<p>二</p>
</li>
</ul>
</blockquote>
```

这是极其违反直觉的。

我们可以要求列表标记（本身也可以缩进）的固定缩进（例如，两个空格，甚至一个空格），而不是要求从边距开始固定缩进，而不必这样做。此提案将消除讨论的最后一个异常。与上面提出的规范不同，它将以下内容算作带有子段落的列表项，即使段落酒吧未缩进至第一段foo：

10. foo

酒吧

可以说，这篇文章读起来确实像一个列表项，酒吧作为小段落，这可能有利于该提案。但是，根据该提案，缩进的代码必须在列表标记后缩进六个空格。这会破坏许多现有的 Markdown，其模式如下：

1. foo

缩进代码

代码缩进 8 个空格。相比之下，上面的规范将按预期解析此文本，因为代码块的缩进是从foo。

需要特殊处理的情况是列表项开始带有缩进的代码。由于我们没有“第一段”作为衡量标准，那么在这种情况下需要多少缩进？规则 #2 只是规定，在这种情况下，我们需要从列表标记开始缩进一个空格（然后是缩进代码的正常四个空格）。在列表标记加上其初始缩进占用四个空格（常见情况）的情况下，这将符合四空格规则，但在其他情况下则有所不同。

## 5.3 列表

一个列表 是一个或多个列表项的序列同一类型。列表项可以由任意数量的空行分隔。

两个列表项是同一类型 如果他们以列表标记 属于同一类型。如果 (a) 两个列表标记是使用相同字符 (-、+ 或 \*) 的项目符号列表标记，或 (b) 它们是使用相同分隔符 (.) 或 () 的有序列表编号，则这两个列表标记属于同一类型。

列表是有序列表 如果其组成列表项以有序列表标记，以及项目符号列表 如果其组成列表项以项目符号列表标记。

这起始编号 的有序列表 由其首列表项的列表编号决定。后续列表项的编号将被忽略。

列表是松动的 如果其组成列表项之间有空行分隔，或者其组成列表项直接包含两个块级元素，且中间有空行。否则列表为紧密的 (HTML 输出的区别在于，松散列表中的段落被包裹在<对标签，而紧密列表中的段落则不是。)

更改项目符号或有序列表分隔符将开始一个新列表：

### 示例 301

- 富
- 酒吧
- + 巴兹

```
<ul>
<li>foo</li>
<li>酒吧</li>
</ul>
<ul>
<li>巴兹</li>
</ul>
```

### 示例 302

1. •富
2. •酒吧
- 3) 巴兹

```
<ol>
<li>foo</li>
<li>酒吧</li>
</ol>
<ol *开始=“3” >
<li>巴兹</li>
</ol>
```

在 CommonMark 中，列表可以打断段落。也就是说，不需要空行来将段落与后面的列表分开：

### 示例 303

福  
- 酒吧  
- 巴兹

```
<p>食物</p>
<ul>
<li>酒吧</li>
<li>巴兹</li>
</ul>
```

Markdown.pl不允许这样做，因为担心通过硬换行中的数字触发列表：

我家有 14 扇窗户，6 扇门。

但奇怪的是，Markdown.pl *做*允许块引用打断段落，尽管可能适用相同的考虑。

在 CommonMark 中，我们允许列表打断段落，原因有二。首先，人们在列表开头不留空行是很自然且常见的：

我需要买  
- 新鞋  
- 一件外套  
- 一张机票

其次，我们被一个

**统一性原则**：如果一段文本具有某种含义，那么当将其放入容器块（例如列表项或块引用）时，它将继续具有相同的含义。

（事实上，[列表项](#) 和 [块引用](#) 以这个原则为前提。）这个原则意味着，如果

```
* 我需要购买
- 新鞋
- 一件外套
- 一张机票
```

是一个包含段落和嵌套子列表的列表项，所有 Markdown 实现都同意这一点（尽管该段落可以在没有 <对标签，因为列表很“紧”），那么

我需要买  
- 新鞋  
- 一件外套  
- 一张机票

其本身应该是一个段落，后面跟着一个嵌套的子列表。

由于 Markdown 的惯例是允许列表打断列表项内的段落，因此 [统一性原则](#) 要求我们也允许这个外部列表项。（[重新结构化文本](#) 采用不同的方法，要求列表前有空行，即使在其他列表项内也是如此。）

为了解决段落中数字换行的问题，我们只允许以1打断段落。因此，

#### [示例 304](#)

这 · 数字 · 的 视窗 · 在 我的房子 · 是  
14 · · 这 · 数字 · 的 门 · 是 6.

```
<p>· 数字 · 的 视窗 · 在 我的房子 · 是
14 · · 这 · 数字 · 的 门 · 是 6.</p>
```

我们仍然可能会在以下情况下得到意想不到的结果

#### [示例 305](#)

这 · 数字 · 的 视窗 · 在 我的房子 · 是  
1. · 这 · 数字 · 的 门 · 是 6.

```
<p>· 数字 · 的 视窗 · 在 我的房子 · 是</p>
<ol>
<li>· 数字 · 的 门 · 是 6.</li>
</ol>
```

但这条规则应该可以防止大多数虚假列表捕获。

项目之间可以有任意数量的空行：

#### [示例 306](#)

- 富 <ul><li>
  - 酒吧 <p>foo</p></li><li>
  - 巴兹 <p>酒吧</p></li><li>

### 示例 307

- ```
- 富 <ul>
  . - 酒吧 <li>foo
  . . - 巴兹 <ul>
    . . . 比姆 <li>巴兹</p>
    . . . 比姆 <li>bim</p>
  </li>
</ul>
</li>
</ul>
</li>
</ul>
</li>
</ul>
```

为了分隔相同类型的连续列表，或将列表与缩进的代码块分开（否则将被解析为最终列表项的子段落），您可以插入空格 HTML 注释：

### 示例 308

- ```
- 富 <ul><li>foo</li>
- 酒吧</li></ul>
<!-- * -->
<!-- * -->
- 巴兹 <ul><li>巴兹</li>
- 比姆<li>比姆</li>
</ul>
```

### 示例 309

- |            |                            |
|------------|----------------------------|
| - . . 富    | <ul>                       |
| . . . 不编码  | <li><p>foo</p>             |
| - . . 富    | <li><p>没有代码</p>            |
| <!-- * --> | </li>                      |
| . . . 代码   | <li><p>foo</p>             |
|            | </li>                      |
|            | </ul>                      |
|            | <!-- * -->                 |
|            | <pre><code>代码</code></pre> |

列表项不需要缩进到同一级别。以下列表项将被视为同一列表级别的项，因为它们的缩进程度不足以归属于上一个列表项：

### 示例 310

- 一个
    - b
    - 丙
    - d
    - 埃
    - f
  - 克

## [示例 311](#)

- 1. •一个
- 2. •b
- 3. •丙

```
<ol>
<li>
<p>—个</p>
</li>
<li>
<p>二</p>
</li>
<li>
<p><p></p>
</li>
</ol>
```

但请注意，列表项前面的缩进不能超过三个空格。这里 - 埃被视为段落续行，因为它的缩进量超过三个空格：

## [示例 312](#)

- •一个
- b
- 丙
- .. d
- ... 埃

```
<ul>
<li>—个</li>
<li>乙</li>
<li>c</li>
<li>d
- 嗯</li>
</ul>
```

这里，3.c被视为缩进的代码块，因为它缩进四个空格并且前面有一个空行。

## [示例 313](#)

- 1. •一个
- 2. •b
- .. 3. •丙

```
<ol>
<li>
<p>—个</p>
</li>
<li>
<p>二</p>
</li>
</ol>
<pre><code>3. • 丙
</代码></ pre>
```

这是一个松散的列表，因为两个列表项之间有一个空行：

## [示例 314](#)

- •一个
- b
- 丙

```
<ul>
<li>
<p>—个</p>
</li>
<li>
<p>二</p>
</li>
<li>
<p><p></p>
</li>
</ul>
```

这是这样的，第二个项目是空的：

## [示例 315](#)

```
* 一个
*
* 丙


- <ul>
    <li><p>一个</p>
    </li>
    <li><p>丙</p>
    </li>
</ul>

```

这些都是松散的列表，即使项目之间没有空行，因为其中一个项目直接包含两个块级元素，它们之间有一个空行：

#### 示例 316

```
- 一个
- b
- 丙
- d


- <ul>
    <li><p>一个</p>
    </li>
    <li><p>二</p>
        <p>参考: /网址</p>
    </li>
    <li><p>三</p>
    </li>
</ul>

```

#### 示例 317

```
- 一个
- b
- [参考]: /网址
- d


- <ul>
    <li><p>一个</p>
    </li>
    <li><p>二</p>
    </li>
    <li><p>三</p>
    </li>
</ul>

```

这是一个紧密的列表，因为空白行位于代码块中：

#### 示例 318

```
- 一个
- . .
- b.
- 丙


- <ul>
    <li><code>b</code>
    </li>
    <li><code>c</code>
    </li>
</ul>

```

这是一个紧密列表，因为空行位于子列表的两个段落之间。因此子列表是松散的，而外部列表是紧密的：

#### 示例 319

```
- 一个
.. .b
...
.. 丙
- d

<ul>
<li>一个
<ul>
<li>
<p>二</p>
<p><p></p>
</li>
</ul>
</li>
<li>d</li>
</ul>
```

这是一个紧凑的列表，因为空行位于块引用内：

#### [示例 320](#)

```
* 一个
. > .b
. > .
* 丙

<ul>
<li>一个
<blockquote>
<p>二</p>
</blockquote>
</li>
<li>c</li>
</ul>
```

这个列表很紧凑，因为连续的块元素没有被空行分隔：

#### [示例 321](#)

```
- 一个
. > .b
. . .
. 丙
. . .
- d

<ul>
<li>一个
<blockquote>
<p>二</p>
</blockquote>
<pre><code>c
</code></pre>
</li>
<li>d</li>
</ul>
```

单段列表非常紧凑：

#### [示例 322](#)

```
- 一个

<ul>
<li>一个</li>
</ul>
```

#### [示例 323](#)

```
- 一个
.. .b

<ul>
<li>一个
<ul>
<li>乙</li>
</ul>
</li>
</ul>
```

这个列表比较松散，因为列表项中的两个块元素之间有一个空行：

#### [示例 324](#)

```
1. . .
. . 富
. . .
. . 酒吧

<ol>
<li>
<pre><code>foo
</code></pre>
<p>酒吧</p>
</li>
</ol>
```

这里外部列表比较松散，而内部列表比较紧密：

## 示例 325

- \* 富
- \* 酒吧
- 巴兹

```
<ul>
<li>
<p>foo</p>
<ul>
<li>酒吧</li>
</ul>
<p>巴兹</p>
</li>
</ul>
```

## 示例 326

- 一个
- - b
- - 丙

  

- d
- - 埃
- - f

```
<ul>
<li>
<p>一个</p>
<ul>
<li>乙</li>
<li>c</li>
</ul>
</li>
<li>
<p></p>
<ul>
<li>电子</li>
<li>f</li>
</ul>
</li>
</ul>
```

## 6 内联

内联按字符流的开头到结尾的顺序进行解析（在从左到右的语言中，从左到右）。因此，例如，在

## 示例 327

`嗨` 咯`

```
<p><code>嗨</code>嗨`</p>
```

你好被解析为代码，将末尾的反引号保留为文字反引号。

### 6.1 代码跨度

一个反引号字符串是一个或多个反引号(`)的字符串，其前面和后面均没有反引号。

一个代码跨度以反引号字符串开头，以等长反引号字符串结尾。代码跨度的内容是这两个反引号字符串之间的字符，按以下方式规范化：

- 第一的，行尾 转换为空格。
- 如果结果字符串都以和以空间 性格，但并不完全由空间 字符，单个空间 字符会从前面和后面移除。这样您就可以包含以反引号字符开头或结尾的代码，这些代码必须用空格与开头或结尾的反引号字符串分隔开。

这是一个简单的代码跨度：

## 示例 328

`foo`

```
<p><code>foo</code></p>
```

这里使用了两个反引号，因为代码中包含一个反引号。此示例还演示了如何删除单个前导空格和尾随空格：

## 示例 329

`` 富 · `` 酒吧 · ``

```
<p><code>foo · ``</code></p>
```

此示例显示了删除前导和尾随空格的动机：

## [示例 330](#)

```
<p><code>` `</code></p>
```

请注意，只有一空间被剥离：

## [示例 331](#)

```
<p><code> . ` . </code></p>
```

仅当字符串两侧都有空格时才会进行剥离：

## [示例 332](#)

```
<p><code> . ` . </code></p>
```

仅有的空格，而不是unicode 空格一般情况下，都是这样剥离的：

## [示例 333](#)

```
<p><code> b </code></p>
```

如果代码范围仅包含空格，则不会发生剥离：

## [示例 334](#)

```
<p><code> </code>
<code> . . </code></p>
```

行尾 被视为空格：

## [示例 335](#)

```
<p><code>foo *酒吧 . . </code></p>
```

```
富
酒吧 . .
巴兹
```

## [示例 336](#)

```
<p><code>foo . </code></p>
```

```
富 .
```

内部空间未折叠：

## [示例 337](#)

```
<p><code>foo . . 酒吧 </code></p>
```

```
巴兹`
```

请注意，浏览器在渲染<时通常会折叠连续的空格代码>元素，因此建议使用以下 CSS：

```
代码{white-space: 预包装; }
```

请注意，反斜杠转义在代码范围内不起作用。所有反斜杠都按字面意思处理：

## [示例 338](#)

```
<p><code>foo\`bar` 酒吧` </code></p>
```

反斜杠转义是不需要的，因为人们总是可以选择一串n反引号字符作为分隔符，其中代码不包含任何完全相同的字符串n反引号字符。

## [示例 339](#)

``foo`bar``

<p><code>foo`bar</code></p>

### 示例 340

`富。` 酒吧。`

<p><code>foo 。` 酒吧。`</p>

代码跨度反引号的优先级高于除 HTML 标记和自动链接之外的任何其他内联构造。因此，例如，以下代码不会被解析为强调文本，因为第二个 \* 是代码跨度的一部分：

### 示例 341

\* foo`\*`

<p>\*foo<code>\*</code></p>

这不会被解析为链接：

### 示例 342

[不是一个 · 链接](/foo`)

<p>[不 · 一个<code>链接] (/ foo </ code>) </ p>

代码跨度、HTML 标记和自动链接具有相同的优先级。因此，这是代码：

### 示例 343

`<`>`

<p><code><a . href="`>">`</p>

但这是一个 HTML 标签：

### 示例 344

<`>`

<p><a href="`>">`</p>

这是代码：

### 示例 345

`<https://foo.bar.` baz>`

<p><code><https://foo.bar.</code>baz>`</p>

但这是一个自动链接：

### 示例 346

<https://foo.bar.` baz>`

<p><a .  
href="https://foo.bar.%60baz">https://foo.bar.` baz </a>`</p>

当反引号字符串没有被匹配的反引号字符串关闭时，我们只使用文字反引号：

### 示例 347

```foo``

<p>```foo``</p>

### 示例 348

`foo

<p>`foo</p>

以下案例也说明了开始和结束反引号字符串的长度需要相等：

### 示例 349

`foo``bar``

<p>`foo<code>bar</code></p>

## 6.2 强调和强烈强调

约翰·格鲁伯的原作[Markdown 语法说明](#)说：

Markdown 将星号 (\*) 和下划线 (\_) 视为强调符号。用一个 \* 或 \_ 包裹的文本将被 HTML <em> 标签；双 \* 或 \_ 将被 HTML <strong> 标签。

对于大多数用户来说，这已经足够了，但这些规则留下了很多未决问题，尤其是在嵌套强调方面。原始 Markdown.pl 测试套件清楚地表明，三重 \*\*\* 和 \_\_\_ 分隔符可用于强烈强调，并且大多数实现也允许以下模式：

```
*** 强烈强调***  
*** 强烈**强调*  
*** 强调* 强烈**  
* 强烈 *强调***  
* 强调 **强烈***
```

以下模式未得到广泛支持，但其意图很明确并且很有用（特别是在参考书目条目之类的上下文中）：

```
* emph *带有 emph* *  
** 强劲**，实力雄厚**
```

许多实现还将单词内的强调限制为 \* 形式，以避免在包含内部下划线的单词中出现不必要的强调。（最佳做法是将它们放在代码跨度中，但用户通常不这样做。）

内部强调：foo\*bar\*baz 无强调：  
foo\_bar\_baz

下面给出的规则捕获了所有这些模式，同时允许不回溯的有效解析策略。

首先，一些定义。[分隔符运行](#) 是一个或多个 \* 字符的序列，其前面或后面均没有非反斜杠转义的 \* 字符，或者是一个或多个 \_ 字符的序列，其前面或后面均没有非反斜杠转义的 \_ 字符。

一个[左侧分隔符](#) 是分隔符运行 即 (1) 不跟[Unicode 空格](#) 并且 (2a) 后面不跟[Unicode 标点符号](#) 或 (2b) 后跟[Unicode 标点符号](#) 之前[Unicode 空格](#) 或[Unicode 标点符号](#) . 为了本定义的目的，行的开始和结束都算作 Unicode 空格。

一个[右侧分隔符运行](#) 是分隔符运行 即 (1) 之前没有[Unicode 空格](#) 并且 (2a) 前面不加[Unicode 标点符号](#) 或 (2b) 前面加[Unicode 标点符号](#) 然后[Unicode 空格](#) 或[Unicode 标点符号](#) . 为了本定义的目的，行的开始和结束都算作 Unicode 空格。

以下是一些分隔符的示例。

- 左侧翼，但不右侧翼：

```
*** 美国广播公司  
_abc  
** “abc”  
_ “abc”
```

- 右侧翼侧但不左侧翼侧：

```
ABC***  
ABC_  
“abc” **  
“abc”
```

- 左侧和右侧：

```
ABC***DEF  
“abc” _ “def”
```

- 既不偏左也不偏右：

```
abc *** def  
—_—
```

(根据前后字符区分左侧和右侧分隔符的想法来自 Roopesh Chander 的[频率调制](#)。vfmd 使用术语“强调指示字符串”而不是“分隔符运行”，并且其区分左侧和右侧运行的规则比此处给出的规则稍微复杂一些。)

以下规则定义强调和强烈强调：

1. 一个 \* 字符[可以打开强调](#) 当且仅当它是[左侧分隔符](#)。
2. 一个 \_ 字符[可以打开强调](#) 当且仅当它是[左侧分隔符](#) 并且 (a) 不属于[右侧分隔符运行](#) 或 (b) 部分[右侧分隔符运行](#) 前面有一个[Unicode 标点符号](#)。
3. 一个 \* 字符[可以结束强调](#) 当且仅当它是[右侧分隔符运行](#)。
4. 一个 \_ 字符[可以结束强调](#) 当且仅当它是[右侧分隔符运行](#) 并且 (a) 不属于[左侧分隔符](#) 或 (b) 部分[左侧分隔符](#) 随后是[Unicode 标点符号](#)。
5. 双\*\*[可以打开强调](#) 当且仅当它是[左侧分隔符](#)。
6. 双 \_\_[可以打开强调](#) 当且仅当它是[左侧分隔符](#) 并且 (a) 不属于[右侧分隔符运行](#) 或 (b) 部分[右侧分隔符运行](#) 前面有一个[Unicode 标点符号](#)。
7. 双\*\*[可以关闭强烈强调](#) 当且仅当它是[右侧分隔符运行](#)。
8. 双 \_\_[可以关闭强烈强调](#) 当且仅当它是[右侧分隔符运行](#) 并且 (a) 不属于[左侧分隔符](#) 或 (b) 部分[左侧分隔符](#) 随后是[Unicode 标点符号](#)。
9. 强调以分隔符开始，[可以打开强调](#) 并以分隔符结尾[可以结束强调](#)，并使用相同的字符（\_ 或 \*）作为开始分隔符。开始和结束分隔符必须属于单独的[分隔符运行](#)。如果其中一个分隔符既可以打开又可以关闭强调，则包含打开和关闭分隔符的分隔符序列的长度总和不能是 3 的倍数，除非两个长度都是 3 的倍数。
10. 强调语从以下分隔符开始：[可以打开强调](#) 并以分隔符结尾[可以关闭强烈强调](#)，并使用相同的字符（\_ 或 \*）作为开始分隔符。开始和结束分隔符必须属于单独的[分隔符运行](#)。如果其中一个分隔符既可以打开又可以关闭强调，则包含打开和关闭分隔符的分隔符序列的长度总和不能是 3 的倍数，除非两个长度都是 3 的倍数。
11. 文字 \* 字符不能出现在 \* 分隔的强调或 \*\* 分隔的强强调的开头或结尾，除非使用反斜杠转义。
12. 文字 \_ 字符不能出现在 \_ 分隔的强调或 \_\_ 分隔的强强调的开头或结尾，除非使用反斜杠转义。

当上述规则 1-12 与多种解析兼容时，以下原则可以解决歧义问题：

13. 嵌套的次数应尽量减少。例如，解释 <strong> ... </strong> 总是优先于 <em><em>...</em></em>。
14. 解释<><strong>...</strong></em>总是优先于 <><strong>...</em></strong>.
15. 当两个潜在强调或强强调跨度重叠时，即第二个强调在第一个强调结束之前开始，在第一个强调结束之后结束，则第一个强调优先。因此，例如， \*foo \_bar\* baz\_ 解析为 <em>foo \_bar</em> baz\_ 而不是 \*foo <em>酒吧\* baz</em>。
16. 当两个潜在强调或强强调跨度具有相同的结束分隔符时，较短的那个（较晚打开的那个）优先。因此，例如， \*\*foo \*\*酒吧巴兹\*\* 解析为 \*\*foo <strong>酒吧巴兹</strong> 而不是 <strong>foo \*\*bar baz</strong>。
17. 内联代码跨度、链接、图像和 HTML 标签的分组比强调更紧密。因此，当在包含这些元素之一的解释和不包含这些元素的解释之间做出选择时，前者总是胜出。因此，例如， \*[foo\*](酒吧) 解析为 \*<a href="bar">foo\*</a> 而不是 <em>[foo</em>](酒吧)。

这些规则可以通过一系列例子来说明。

规则 1：

## 示例 350

\* foo 酒吧\*

<p><em>foo ·</em></p>

这不是强调，因为开头的 \* 后面是空格，因此不是左侧分隔符：

## 示例 351

一个富·酒吧\*

<p>一个\* 富·酒吧\*</p>

这不是强调，因为开头的 \* 前面是字母数字，后面是标点符号，因此不是左侧分隔符：

## 示例 352

一个\* “foo” \*

<p>a\* "foo" \*</p>

Unicode 不间断空格也算作空格：

## 示例 353

\*一个\*

<p>\* 一个\*</p>

Unicode 符号也算作标点符号：

## 示例 354

\* \$\*alpha。

<p>\*\$\*alpha.</p>

\*£\*太棒了。

<p>\*£\*太棒了。</p>

\* 欧元\*查理。

允许使用 \* 来强调单词：

## 示例 355

foo\*酒吧\*

<p>foo<em>酒吧</em></p>

## 示例 356

5\*6\*78

<p>5<em>6</em>78</p>

规则2：

## 示例 357

\_foo 酒吧\_

<p><em>foo ·</em></p>

这不是强调，因为开头的 \_ 后面是空格：

## 示例 358

\_ 富·酒吧\_

<p>\_ ·富·酒吧\_</p>

这不是强调，因为开头的 \_ 前面是字母数字，后面是标点符号：

## 示例 359

一个 “foo”

<p>a\_ "foo" \_</p>

单词内不允许使用 \_ 来强调：

## 示例 360

foo\_bar\_

<p>foo\_bar\_</p>

### 示例 361

5\_6\_78

<p>5\_6\_78</p>

### 示例 362

пристаням\_стремятся\_

<p>пристаням\_стремятся\_</p>

这里 \_ 不产生强调作用，因为第一个分隔符位于右侧，而第二个分隔符位于左侧：

### 示例 363

aa\_ "bb" \_cc

<p>aa\_ "bb" \_cc</p>

这就是强调，尽管开头的分隔符位于左侧和右侧，因为它前面有标点符号：

### 示例 364

foo\_-(酒吧)\_

<p>foo-<em>(bar)</em></p>

规则 3：

这不是强调，因为结束定界符与开始定界符不匹配：

### 示例 365

\_foo\*

<p>\_foo\*</p>

这不是强调，因为结尾的 \* 前面有空格：

### 示例 366

\* foo ·酒吧\*

<p>\*foo ·酒吧\*</p>

行尾也算作空格：

### 示例 367

\* foo ·酒吧  
\*

<p>\*foo ·酒吧  
\*</p>

这不是强调，因为第二个 \* 前面是标点符号，后面是字母数字（因此它不是右侧分隔符运行：

### 示例 368

\* (\*foo)

<p>\*(\*foo)</p>

通过这个例子可以更容易地理解这个限制的意义：

### 示例 369

\* (\*foo\*) \*

<p><em>(<em>foo</em>)</em></p>

允许使用 \* 来强调单词：

### 示例 370

\* foo \* bar

<p><em>foo</em>酒吧</p>

规则4：

这不是强调，因为结束的 \_ 前面有空格：

### 示例 371

\_foo ·酒吧\_·

<p> \_foo ·酒吧\_</p>

这不是强调，因为第二个 \_ 前面是标点符号，后面是字母数字：

### 示例 372

\_ (\_foo)\_

<p>\_ (\_foo) </p>

这是强调中的强调：

### 示例 373

\_ (\_foo\_) \_

<p><em>(<em>foo</em>)</em></p>

不允许对 \_ 进行词内强调：

### 示例 374

\_foo\_bar

<p>\_foo\_bar</p>

### 示例 375

\_пристаням\_стремятся

<p>\_пристаням\_стремятся</p>

### 示例 376

\_foo\_bar\_baz\_

<p><em>foo\_bar\_baz</em></p>

这就是强调，尽管结束分隔符位于左侧和右侧，因为它后面跟着标点符号：

### 示例 377

\_ (酒吧) \_。

<p><em>(酒吧)</em>. </p>

规则 5：

### 示例 378

\*\* 富 · 酒吧\*\*

<p><strong>foo · </strong></p>

这不是强烈强调，因为开头的分隔符后面是空格：

### 示例 379

\*\* 富 · 酒吧\*\*

<p>\*\* ·富 · 酒吧\*\*</p>

这不是强烈强调，因为开头的 \*\* 前面是字母数字，后面是标点符号，因此不是左侧分隔符：

### 示例 380

啊\*\* "foo" \*\*

<p>a\*\*"foo"\*\*</p>

允许在单词内使用 \*\* 进行强调：

### 示例 381

foo\*\*酒吧\*\*

<p>foo<strong>酒吧</strong></p>

规则6：

## 示例 382

\_\_foo ·酒吧\_\_

<p><strong>foo · </strong></p>

这不是强烈强调，因为开头的分隔符后面是空格：

## 示例 383

\_\_ 富 · 酒吧\_\_

<p>\_\_ ·富 · 酒吧\_\_</p>

行尾算作空格：

## 示例 384

富 · 酒吧\_\_

<p>\_\_  
富 · 酒吧\_\_</p>

这不是强烈强调，因为开头的 \_\_ 前面是字母数字，后面是标点符号：

## 示例 385

一个 "foo" \_\_

<p>a\_\_ "foo" \_\_</p>

禁止在单词内使用 \_\_ 来强调：

## 示例 386

foo\_bar\_\_

<p>foo\_bar\_\_</p>

## 示例 387

5\_6\_78

<p>5\_6\_78</p>

## 示例 388

пристаням\_\_ стремятся\_\_

<p>пристаням\_\_ стремятся\_\_</p>

## 示例 389

\_\_foo, ·\_\_酒吧\_\_, ·巴兹\_\_

<p><strong>foo, · <strong>酒吧</strong>, ·巴兹</p>

尽管开头的分隔符位于左侧和右侧，但这是强烈强调，因为它前面有标点符号：

## 示例 390

foo-\_\_(酒吧)\_\_

<p>foo-<strong> (酒吧) </strong></p>

规则 7：

这不是强烈强调，因为结束分隔符前面有空格：

## 示例 391

\*\* 富 · 酒吧\*\*·\*

<p>\*\*foo · 酒吧\*\* ·\*</p>

(也不能理解为强调\*foo 样 \*，因为规则 11。)

这不是强烈强调，因为第二个 \*\* 前面是标点符号，后面是字母数字：

## 示例 392

\*\* (\*\*foo)

<p>\*\* (\*\*foo) </p>

通过以下示例可以更容易理解此限制的意义：

### 示例 393

\* (\*\*foo\*\*) \* <p><em>(<strong>foo</strong>)</em></p>

### 示例 394

\* \* 嵌果木属 · (\* 嵌果木 · 风箱树属\*) \* · 同义词。 <p><strong>嵌果木属 · (<em>Gomphocarpus · 风箱果属</em>), · 同义词。 马利筋 · </p>

### 示例 395

\*\* 富 · “\*酒吧\*” .foo\*\* <p><strong>foo · “<em>酒吧</em>” · </strong></p>

词内强调：

### 示例 396

\*\*foo\*\*bar <p><strong>foo</strong>酒吧</p>

规则 8：

这不是强烈强调，因为结束分隔符前面有空格：

### 示例 397

\_foo ·酒吧\_ <p>\_foo ·酒吧\_</p>

这不是强烈强调，因为第二个 \_ 前面是标点符号，后面是字母数字：

### 示例 398

\_ (\_foo) <p>\_(\_foo)</p>

通过这个例子可以更容易地理解这个限制的意义：

### 示例 399

(\_foo\_) \_\_ <p><em>(<strong>foo</strong>)</em></p>

禁止在单词内使用 \_\_ 来强调：

### 示例 400

\_foo\_bar <p>\_foo\_bar</p>

### 示例 401

\_пристаням\_\_стремятся <p>\_пристаням\_\_стремятся</p>

### 示例 402

\_foo\_bar\_baz\_\_ <p><strong>foo\_bar\_baz</strong></p>

尽管结束分隔符位于左侧和右侧，但这是强烈强调，因为它后面跟着标点符号：

### 示例 403

\_ (酒吧) \_\_。 <p><strong> (条) </strong>。 </p>

规则 9：

任何非空的内联元素序列都可以作为强调跨度的内容。

## 示例 404

\* foo ·[bar](/url)\* <p><em>foo ·<—href="/url">酒吧</a></em></p>

## 示例 405

\* foo  
酒吧\* <p><em>foo</em></p>

具体来说，强调和强烈强调可以嵌套在强调内部：

## 示例 406

\_foo ·\_酒吧\_·巴兹 <p><em>foo ·<strong>酒吧</strong> · </em></p>

## 示例 407

\_foo \_·酒吧\_·巴兹 <p><em>foo ·<em>酒吧</em> ·</em></p>

## 示例 408

\_·foo\_ ·\_酒吧\_ <p><em><em>foo</em> · </em></p>

## 示例 409

\* foo \* 酒吧\*\* <p><em>foo ·<em>酒吧</em></em></p>

## 示例 410

\* foo \* \* 酒吧\*\* ·巴兹\* <p><em>foo ·<strong>酒吧</strong> · </em></p>

## 示例 411

\*foo\*\*bar\*\*baz\* <p><em>foo<strong>bar</strong>baz</em></p>

请注意，在前面的例子中，解释

<p><em>foo</em><em>bar</em><em>baz</em></p>

被一个既可以打开又可以关闭的分隔符（如后面的 \*）所排除foo)如果包含开始和结束分隔符的分隔字符串的长度之和是 3 的倍数，则不能形成强调，除非两个长度都是 3 的倍数。

出于同样的原因，我们在这个例子中没有得到两个连续的强调部分：

## 示例 412

\*foo\*\*酒吧\* <p><em>foo\*\*bar</em></p>

相同的条件确保以下情况都是强调嵌套在强调内，即使省略了内部空格：

## 示例 413

\*\*\* 富\*\* ·酒吧\* <p><em><strong>foo</strong> · </em></p>

## 示例 414

\* foo \* \* 酒吧\*\*\* <p><em>foo ·<strong>酒吧</strong></em></p>

## 示例 415

\* foo\*\*酒吧\*\*\* <p><em>foo<strong>酒吧</strong></em></p>

当内部结束和开始分隔符的长度为两个都但是，3 的倍数可以匹配以产生强调：

## 示例 416

砰砰砰

<p>foo<em><strong>bar</strong></em>baz</p>

## 示例 417

foo\*\*\*\*\*bar\*\*\*\*\*baz

<p>foo<strong><strong><strong>bar</strong></strong></strong>\*\*\*baz</p>

可以实现无限级别的嵌套：

## 示例 418

\* foo \*\* 酒吧 \*巴兹\* ·比姆\*\* ·波普\*

<p><em>foo ·<strong>酒吧 \*<em>巴兹</em> ·</strong> </strong> ·</em></p>

## 示例 419

\* foo ·[\*bar\*](/url)\*

<p><em>foo ·<!--href="/url"--><em>栏</em></a></em></p>

不能有空洞的强调或强烈的强调：

## 示例 420

\*\* 是 不是一个空的 · 强调

<p>\*\* ·是 不是一个空的 · 强调 </p>

## 示例 421

\*\*\*\* 是 不是一个空的 · 强的 · 强调

<p>\*\*\*\* ·是 不是一个空的 · 强的 · 强调 </p>

规则 10：

任何非空的内联元素序列都可以是强调跨度的内容。

## 示例 422

\*\* 富 · [bar] (/url) \*\*

<p><strong>foo · <!--href="/url"-->酒吧</a></strong></p>

## 示例 423

\*\* 富  
酒吧\*\*

<p><strong>foo

具体来说，强调和强调可以嵌套在强调之中：

## 示例 424

\_foo ·\_酒吧\_ ·巴兹\_

<p><strong>foo · <em>酒吧</em> ·</strong></p>

## 示例 425

\_foo ·\_酒吧\_ ·巴兹\_

<p><strong>foo · <strong>酒吧</strong> · 巴兹</p>

## 示例 426

\_\_\_foo\_\_ ·酒吧\_\_

<p><strong><strong>foo</strong> ·</strong></p>

## 示例 427

\*\* 富 · \*\* 酒吧\*\*\*\*

<p><strong>foo · <strong>酒吧</strong></strong></p>

## 示例 428

\*\* 富 · \* 酒吧\* ·巴兹\*\*

<p><strong>foo · <em>酒吧</em> ·</strong></p>

## 示例 429

\* \* foo\*bar\*baz\*\*

<p><strong>foo<em>bar</em>baz</strong></p>

## 示例 430

\*\*\* 富\* •酒吧\*\*

<p><strong><em>foo</em> • </strong></p>

### 示例 431

\*\* 富 \* 酒吧\*\*\*

<p><strong>foo \* <em>酒吧</em></strong></p>

可以实现无限级别的嵌套：

### 示例 432

\*\* 富 \* 酒吧 \*\* 巴兹\*\*  
bim\* 波普\*\*

<p><strong>foo \* <em>酒吧 <strong>巴兹</strong></em> </em> </strong></p>

### 示例 433

\*\* 富 [\*bar\*](/url)\*\*

<p><strong>foo \* <a href="/url"><em>酒吧</em></a></strong></p>

不能有空洞的强调或强烈的强调：

### 示例 434

\_ 是 不是一个空的 • 强调

<p>\_ ·是 ·不是一个空的 · 强调 ·</p>

### 示例 435

\_\_\_\_ 是 不是一个空的 • 强的 • 强调

<p>\_\_\_\_ ·是 ·不是一个空的 · 强的 · 强调 ·</p>

规则 11：

### 示例 436

富 \* \*\*\*

<p>foo \* \*\*\* </p>

### 示例 437

富 \* \\*\*

<p>foo \* <em>\*</em></p>

### 示例 438

富 \* \_ \*

<p>foo \* <em>\_</em></p>

### 示例 439

富 \* \*\*\*\*\*

<p>foo \* \*\*\*\*\* </p>

### 示例 440

富 \* \* \\*\*\*

<p>foo \* <strong>\*</strong></p>

### 示例 441

富 \* \* \_ \*\*

<p>foo \* <strong>\_</strong></p>

请注意，当分隔符不均匀匹配时，规则 11 确定多余的文字 \* 字符将出现在强调之外，而不是其中：

### 示例 442

\* \* 富\*

<p>\* <em>foo</em></p>

### 示例 443

\* foo\*\*

<p><em>foo</em>\*</p>

### 示例 444

\* \* \* 富\*\*

<p>\* <strong>foo</strong></p>

### 示例 445

\*\*\* 富\*

<p>\*\*\*<em>foo</em></p>

## 示例 446

\*\* 啊\*\*

<p><strong>foo</strong>\*</p>

## 示例 447

\* \*\*\*\*

<p><em>foo</em>\*\*\*</p>

规则 12:

## 示例 448

富 • \_\_

<p>foo •\_\_</p>

## 示例 449

富 • \_\_

<p>foo •<em>\_</em></p>

## 示例 450

富 • \_•

<p>foo •<em>\*</em></p>

## 示例 451

富 • \_\_\_\_\_

<p>foo •\_\_\_\_\_</p>

## 示例 452

富 • \_\_\_\_\_

<p>foo •<strong>\_</strong></p>

## 示例 453

富 • \_\*\_\_

<p>foo •<strong>\*</strong></p>

## 示例 454

\_foo\_

<p>\_<em>foo</em></p>

请注意，当分隔符不均匀匹配时，规则 12 确定多余的文字 \_ 字符将出现在强调之外，而不是其中：

## 示例 455

\_foo\_

<p><em>foo</em>\_</p>

## 示例 456

\_\_\_foo\_\_

<p>\_<strong>foo</strong></p>

## 示例 457

\_\_\_\_foo\_\_

<p>\_\_<em>foo</em></p>

## 示例 458

\_foo\_\_

<p><strong>foo</strong>\_</p>

## 示例 459

\_\_\_foo\_\_

<p><em>foo</em>\_\_</p>

规则 13 意味着如果您希望强调直接嵌套在强调内，则必须使用不同的分隔符：

## 示例 460

\*\* 富\*\*

<p><strong>foo</strong></p>

## 示例 461

\* \_foo\_ \*

<p><em><em>foo</em></em></p>

## 示例 462

\_foo\_

<p><strong>foo</strong></p>

## 示例 463

\_ \*foo\* \_

<p><em><em>foo</em></em></p>

但是，无需切换分隔符，就可以在强调内进行强调：

## 示例 464

\*\*\*\* foo\*\*\*\*

<p><strong><strong>foo</strong></strong></p>

## 示例 465

\_\_\_\_foo\_\_\_\_\_

<p><strong><strong>foo</strong></strong></p>

规则 13 可应用于任意长的定界符序列：

## 示例 466

\*\*\*\*\* foo\*\*\*\*\*

<p><strong><strong><strong>foo</strong></strong></p>

规则 14：

## 示例 467

\*\*\* 碰碰碰

<p><em><strong>foo</strong></em></p>

## 示例 468

\_\_\_\_foo\_\_\_\_\_

<p><em><strong><strong>foo</strong></strong></em></p>

规则 15：

## 示例 469

\* foo \* \_酒吧\* ·巴茲

<p><em>foo \* \_bar</em> \* </p>

## 示例 470

\* foo \* \_酒吧\* ·巴茲\* · 碰\*

<p><em>foo \* <strong>酒吧 \* 巴茲</strong> 碰</em></p>

规则 16：

## 示例 471

\*\* 富 \* \*\* 酒吧 ·巴茲\*\*

<p>\*\*foo \* <strong>酒吧 \* 巴茲</strong></p>

## 示例 472

\* foo \* 酒吧 ·巴茲\*

<p>\*foo \* <em>酒吧 </em></p>

规则 17：

## 示例 473

\* [bar\*](/url)

<p>\*<a href="/url">酒吧\*</a></p>

## 示例 474

\_foo \*[bar\_](/url)

<p>\_foo \* <a href="/url">bar\_</a></p>

## 示例 475

```
* <图片 ·源文件="foo" ·标题="**"/>
<p>*<图片 ·源文件="foo" ·标题="**"/></p>
```

## 示例 476

```
**<-- href="***">
<p>**<a ·href="***"></p>
```

## 示例 477

```
_<-- href="__">
<p>_<a ·href="__"></p>
```

## 示例 478

```
*一个 · *` *
<p><em>一个 <code>*</code></em></p>
```

## 示例 479

```
_一个 · __
<p><em>一个 <code>_</code></em></p>
```

## 示例 480

```
** a<https://foo.bar/?q=**>
<p>**一个<a href="https://foo.bar/?q=**">https://foo.bar/?q=__</a></p>
```

## 示例 481

```
_a<https://foo.bar/?q=__>
<p>_a<a ·href="https://foo.bar/?q=__">https://foo.bar/?q=__</a></p>
```

## 6.3 链接

链接包含链接文本（可见文本），链接目标（链接目标的 URI），以及可选的链接标题。Markdown 中有两种基本类型的链接。内联链接 目的地和标题紧跟在链接文本之后。在参考链接 目的地和标题在文档的其他地方定义。

一个链接文本 由方括号 ([ 和 ]) 括起来的零个或多个内联元素序列组成。适用以下规则：

- 链接不得包含其他链接，无论嵌套级别如何。如果多个有效的链接定义相互嵌套，则使用最内层的定义。
- 括号内允许链接文本 仅当 (a) 它们是反斜杠转义的，或者 (b) 它们作为一对匹配的括号出现，带有一个左括号 [、零个或多个内联序列和一个右括号 ] 时。
- 反引号代码跨度，自动链接 和原始HTML 标签 比链接文本中的括号绑定得更紧密。因此，例如，[foo`)]` 不能是链接文本，因为第二个] 是代码跨度的一部分。
- 链接文本中的括号比标记绑定得更紧密强调 和 强烈强调。因此，例如，\*[foo\*](网址)是一个链接。

一个链接目标 包括

- 在开头的 < 和结尾的 > 之间包含零个或多个字符的序列，不包含行尾或未转义的 < 或 > 字符，或者
- 不以 < 开头的非空字符序列，不包括ASCII 控制字符 或者空间 字符，并且仅当 (a) 括号是反斜杠转义的或 (b) 括号是匹配的未转义括号对的一部分时才包含括号。（实现可能会对括号嵌套施加限制以避免性能问题，但至少应支持三级嵌套。）

一个链接标题 包括

- 直双引号字符 (" ) 之间的零个或多个字符序列，仅当该字符是反斜杠转义的时才包含 " 字符，或者
- 直单引号字符 (' ) 之间的零个或多个字符序列，仅当 ' 字符以反斜杠转义时才包含该字符，或者

- 匹配括号 ((...)) 之间的零个或多个字符序列，仅当使用反斜杠转义时才包含 ( 或 ) 字符。

虽然链接标题 可能跨越多行，它们可能不包含空行。

一个内联链接 包括一个链接文本 紧接着是左括号 (，可选链接目标，可选链接标题 和右括号 )。这四个部分可以用空格、制表符和最多一个行尾分隔。如果两个链接目标 和链接标题 在场，他们必须由空格、制表符和最多一个行尾分隔。

链接的文本由包含在链接文本（不包括封闭的方括号）。链接的 URI 由链接目标组成，不包括封闭的 <...>（如果存在），反斜杠转义如上所述。链接的标题由链接标题组成，不包括封闭的分隔符，反斜杠转义如上所述。

这是一个简单的内联链接：

#### 示例 482

|                    |                                              |
|--------------------|----------------------------------------------|
| [链接](/uri · “标题” ) | <p><a href="/uri" · title="title">链接</a></p> |
|--------------------|----------------------------------------------|

标题、链接文本甚至目的地都可以省略：

#### 示例 483

|            |                              |
|------------|------------------------------|
| [链接](/uri) | <p><a href="/uri">链接</a></p> |
|------------|------------------------------|

#### 示例 484

|                  |                                  |
|------------------|----------------------------------|
| [] (./target.md) | <p><a href=".target.md"></a></p> |
|------------------|----------------------------------|

#### 示例 485

|         |                          |
|---------|--------------------------|
| [关联] () | <p><a href="">链接</a></p> |
|---------|--------------------------|

#### 示例 486

|          |                          |
|----------|--------------------------|
| [链接](<>) | <p><a href="">链接</a></p> |
|----------|--------------------------|

#### 示例 487

|      |                        |
|------|------------------------|
| []() | <p><a href=""></a></p> |
|------|------------------------|

如果目标括在尖括号中，则只能包含空格：

#### 示例 488

|                 |                   |
|-----------------|-------------------|
| [链接] (/我的 ·uri) | <p>[链接](/我的 ·</p> |
|-----------------|-------------------|

#### 示例 489

|                   |                                   |
|-------------------|-----------------------------------|
| [链接] (</我的 ·uri>) | <p><a href="/my%20uri">链接</a></p> |
|-------------------|-----------------------------------|

目标不能包含行尾，即使用尖括号括起来也是如此：

#### 示例 490

|                 |                       |
|-----------------|-----------------------|
| [链接](foo<br>酒吧) | <p>[链接](foo<br>) </p> |
|-----------------|-----------------------|

#### 示例 491

|                   |                       |
|-------------------|-----------------------|
| [链接](<foo<br>酒吧>) | <p>[链接](<foo><br></p> |
|-------------------|-----------------------|

如果目标括在尖括号中，则可以包含 )：

#### 示例 492

[a](<b>c)

<p><a href="b">c</a></p>

括住链接的尖括号必须取消转义：

#### 示例 493

[链接] (<foo>)

<p>[link](<foo>)</p>

这些不是链接，因为打开的尖括号没有正确匹配：

#### 示例 494

[a](<b>c  
[a](<b>c>  
[a](<b>c)

<p>[a](<b>c  
[a](<b>c>  
[a](<b>c)</p>

链接目标内的括号可以转义：

#### 示例 495

[链接](\((foo\))

<p><a href="(foo)">链接</a></p>

允许使用任意数量的括号，无需转义，只要它们是匹配的：

#### 示例 496

[链接](foo(和(bar)))

<p><a href="foo(and(bar))">链接</a></p>

但是，如果括号不匹配，则需要转义或使用 <...> 形式：

#### 示例 497

[链接] (foo (和 (bar) )

<p>[link](foo(和(bar))</p>

#### 示例 498

[链接](foo\(和\(\bar\)))

<p><a href="foo(and(bar))">链接</a></p>

#### 示例 499

[链接] (<foo (和 (bar) >)

<p><a href="foo(and(bar))">链接</a></p>

括号和其他符号也可以转义，就像在 Markdown 中一样：

#### 示例 500

[链接] (foo\)\:) ;

<p><a href="foo":>链接</a></p>

链接可以包含片段标识符和查询：

#### 示例 501

[链接] (#fragment)

<p><a href="#fragment">链接</a></p>

[链接] (https://example.com#fragment)

<p><a href="https://example.com#fragment">链接</a></p>

[链接] (https://example.com?foo=3#frag)

<p><a href="https://example.com?foo=3#frag">链接</a></p>

请注意，不可转义字符前的反斜杠只是一个反斜杠：

#### 示例 502

[链接](foo\bar)

<p><a href="foo%5Cbar">链接</a></p>

URL 转义应该留在目标中，因为所有 URL 转义字符也是有效的 URL 字符。目标中的实体和数字字符引用将被解析为

和往常一样，它们会对应 Unicode 代码点。当以 HTML 形式编写时，这些代码点可以选择性地进行 URL 转义，但本规范并不强制执行以 HTML 或其他格式呈现 URL 的任何特定策略。渲染器可能会对如何在输出中转义或规范化 URL 做出不同的决定。

### 示例 503

[链接](foo%20bä)

```
<p><a href="foo%20b%C3%A4">链接</a></p>
```

请注意，由于标题通常可以解析为目的地，如果您尝试省略目的地并保留标题，则会得到意外的结果：

### 示例 504

[链接] (“标题”)

```
<p><a href="%22title%22">链接</a></p>
```

标题可以用单引号、双引号或括号括起来：

### 示例 505

[链接] (/url · “标题”)  
[链接] (/url · '标题')  
[链接] (/url · (标题))

```
<p><a href="/网址" · title="title">链接</a>  
<—href="/网址" · title="title">链接</a>  
<—href="/网址" · title="title">链接</a></p>
```

标题中可以使用反斜杠转义符以及实体和数字字符引用：

### 示例 506

[链接] (/url · “标题 \“””)

```
<p><a href="/网址" · title="标题 ·  
""">链接</a></p>
```

标题必须使用空格、制表符和最多一行结尾与链接分开。其他[Unicode 空格](#) 就像不间断空格不起作用一样。

### 示例 507

[链接] (/url “标题”)

```
<p><a href="/url%C2%A0%22title%22">链接</a></p>
```

不允许使用嵌套的平衡引号，除非进行转义：

### 示例 508

[链接] (/url · “标题 · “和” 标题” )

```
<p>[链接] (/url · "标题 ·  
"和" ·  
标题") </p>
```

但通过使用不同的引用类型可以轻松解决这个问题：

### 示例 509

[链接] (/url · '标题 · “和” 标题')

```
<p><a href="/网址" · title="标题 ·  
"和" ·  
标题">链接</a></p>
```

(笔记：Markdown.pl 确实允许在双引号标题中使用双引号，其测试套件中包含一个演示此操作的测试。但很难找到这会带来额外复杂性的合理理由，因为已经有很多方法（反斜杠转义、实体和数字字符引用，或使用不同的引号类型作为封闭标题）来编写包含双引号的标题。Markdown.pl's 对标题的处理还有许多其他奇怪的特点。例如，它允许在内联链接中使用单引号标题，但不允许在引用链接中使用单引号标题。并且，在引用链接中（但不允许在内联链接中）允许标题以“开头并以 结尾。Markdown.pl 1.0.1 甚至允许标题不带右引号，但 1.0.2b8 不允许。似乎最好采用一种简单、合理的规则，该规则在内联链接和链接引用定义中也同样有效。）

目标和标题周围允许使用空格、制表符以及最多一个行尾：

### 示例 510

[关联] (· · · /uri  
· “标题” · ·)

```
<p><a href="/uri" · title="title">链接</a></p>
```

但链接文本和以下括号之间不允许出现：

### 示例 511

[关联] • (/uri)

<p>[链接] • (/uri) </p>

链接文本可以包含匹配的括号，但不能包含不匹配的括号，除非它们被转义：

### 示例 512

[关联 • [foo • [bar]]] (/uri)

<p><a href="/uri">链接。 [foo • [酒吧]]</—></p>

### 示例 513

[关联] • 酒吧] (/uri)

<p>[链接] • </p>

### 示例 514

[关联 • [bar] (/uri)

<p>[链接] • <— href="/uri">酒吧</a></p>

### 示例 515

[关联 • \[bar] (/uri)

<p><a href="/uri">链接。 [酒吧</a></p>

链接文本可能包含内联内容：

### 示例 516

[关联 • \* foo • \* 酒吧\*\* • #`\*] (/uri)

<p><a href="/uri">链接。 <em>foo •  
<strong>酒吧</strong> • <code>#</code></em></a></p>

### 示例 517

[![moon](moon.jpg)] (/uri)

<p><a href="/uri"></p>

但是，在任何嵌套级别中，链接都不能包含其他链接。

### 示例 518

[foo • [酒吧] (/uri) ] (/uri)

<p>[foo • <— href="/uri">酒吧</a>] (/uri) </p>

### 示例 519

[foo • \* [酒吧 • [baz] (/uri) ] (/uri)\*] (/uri)

<p>[foo • <em>[酒吧 • <— href="/uri">baz</a>] (/uri)  
</em>] (/uri)</p>

### 示例 520

![[[foo](uri1)](uri2)](uri3)

<p></p>

这些案例说明了链接文本分组优先于强调分组：

### 示例 521

\* [foo\*] (/uri)

<p>\*<a href="/uri">foo\*</a></p>

### 示例 522

[foo • \* 酒吧] (baz\*)

<p><a href="baz\*">foo • \* 酒吧</a></p>

请注意括号中不是部分链接不优先：

### 示例 523

\* foo • [酒吧\* • 巴兹

<p><em>foo • [酒吧</em> • </p>

这些案例说明了 HTML 标记、代码跨度和自动链接相对于链接分组的优先级：

## 示例 524

```
[foo ·<酒吧属性="](baz)">          <p>[foo ·<酒吧属性="](baz)"></p>
```

## 示例 525

```
[foo` ](/uri)`          <p>[foo<code> ](/uri)</code></p>
```

## 示例 526

```
[foo<https://example.com/?search=](uri)>          <p>[foo<a · href="https://example.com/?search=%5D(uri)">https://example.com/?search=](uri)</a></p>
```

有三种参考链接: 满的, 崩溃, 和捷径。

一个完整参考链接 包括一个链接文本 紧接着链接标签 那火柴 一个链接引用定义 文档的其他地方。

一个链接标签 以左括号 (|) 开头, 以第一个未反斜杠转义的右括号 (|) 结尾。这些括号之间必须至少有一个不是空格、制表符或行尾的字符。未转义的方括号字符不允许出现在链接标签。链接标签在方括号内最多可包含 999 个字符。

一个标签火柴 另一个是为了防止它们的规范化形式相同。要规范化标签, 请去掉开括号和闭括号, 执行 Unicode 大小写折叠, 删除前导和尾随空格、制表符和行尾, 并将连续的内部空格、制表符和行尾折叠为单个空格。如果有多个匹配的引用链接定义, 则使用文档中第一个出现的链接定义。(在这种情况下最好发出警告。)

链接的 URI 和标题由匹配的链接引用定义。

这是一个简单的例子:

## 示例 527

```
[foo][酒吧]          <p><a · href="/网址" · title="title">foo</a></p>  
[酒吧]: ./网址 ·“标题”
```

规则链接文本 与 内联链接。因此:

链接文本可以包含匹配的括号, 但不能包含不匹配的括号, 除非它们被转义:

## 示例 528

```
[关联 ·[foo ·[酒吧]]][参考]          <p><a · href="/uri">链接 · [foo ·[酒吧]]</一></p>  
[参考]: ./uri
```

## 示例 529

```
[关联 ·\[酒吧][参考]          <p><a · href="/uri">链接 · [酒吧]</a></p>  
[参考]: ./uri
```

链接文本可能包含内联内容:

## 示例 530

```
[关联 ·* foo ·* * 酒吧** · #`*][参考]          <p><a · href="/uri">链接 · <em>foo ·  
[参考]: ./uri          <strong>酒吧</strong> · <code>#</code></em></a></p>
```

## 示例 531

```
[![月亮](moon.jpg)][参考]          <p><a · href="/uri"></p>  
[参考]: ./uri
```

但是, 在任何嵌套级别中, 链接都不能包含其他链接。

## 示例 532

[foo · [bar](/uri)][ref] <p>[foo ·<—href="/uri">酒吧</a>][a · href="/uri">参考</a></p>  
[参考]: ./uri

## 示例 533

[foo · \* 酒吧 · [baz][ref]\*][ref] <p>[foo ·<em>酒吧 <—href="/uri">baz</a></em>]<a · href="/uri">参考</a></p>  
[参考]: ./uri

(在上面的例子中，我们有两个[快捷方式参考链接](#) 而不是一个[完整参考链接](#)。

以下案例说明了链接文本分组优于强调分组：

## 示例 534

\* [foo\*][ref] <p>\*<a ·href="/uri">foo\*</a></p>  
[参考]: ./uri

## 示例 535

[foo · \* 酒吧][参考]\* <p><a ·href="/uri">foo · \* 酒吧</a>\*</p>  
[参考]: ./uri

这些案例说明了 HTML 标记、代码跨度和自动链接相对于链接分组的优先级：

## 示例 536

[foo ·<酒吧属性="" ][参考]" > <p>[foo ·<酒吧attr="" ][ref]"></p>  
[参考]: ./uri

## 示例 537

[foo`][ref]` <p>[foo<code>][ref]</code></p>  
[参考]: ./uri

## 示例 538

[foo<https://example.com/?search=][ref]> <p>[foo<a · href="https://example.com/?search=%5D%5Bref%5D">https://example.com/?search=][ref]</a></p>  
[参考]: ./uri

匹配不区分大小写：

## 示例 539

[foo][BaR] <p><a ·href="/网址" · title="标题">foo</a></p>  
[酒吧]: ./网址 ·“标题”

使用 Unicode 大小写折叠：

## 示例 540

[β] <p><a ·href="/url">β</a></p>  
[党卫军]: /网址

为了确定匹配，连续的内部空格、制表符和行尾将被视为一个空格：

## 示例 541

[富  
• 酒吧]: ./网址

<p><a href="/url">巴兹</a></p>

[巴兹][福] • 酒吧]

之间不允许有空格、制表符或行尾链接文本 和链接标签：

#### 示例 542

[foo] • [酒吧]

<p>[foo] • <—href="/网址" • title="title">酒吧</a></p>

[酒吧]: ./网址 •“标题”

#### 示例 543

[foo]  
[酒吧]

<p>[foo]  
<—href="/网址" • title="title">酒吧</a></p>

[酒吧]: ./网址 •“标题”

这与 John Gruber 最初的 Markdown 语法描述不同，后者明确允许在链接文本和链接标签之间留有空格。它使参考链接与内联链接，根据原始 Markdown 和本规范，链接文本后不能有空格。更重要的是，它可以防止无意中捕获连续的快捷方式参考链接。如果允许在链接文本和链接标签之间留有空格，则下面我们将有一个参考链接，而不是两个快捷参考链接，正如预期的那样：

[foo]  
[酒吧]

[foo]: /url1  
[栏]: /url2

(注意快捷方式参考链接 由格鲁伯本人在测试版中介绍Markdown.pl，但从未包含在官方语法描述中。如果没有快捷引用链接，在链接文本和链接标签之间留出空格是无害的；但一旦引入快捷引用，允许这样做就太危险了，因为它经常会导致意想不到的结果。)

当有多个匹配时链接参考定义，第一个被使用：

#### 示例 544

[foo]: ./url1

<p><a href="/url1">酒吧</a></p>

[foo]: ./url2

[酒吧][foo]

请注意，匹配是在规范化的字符串上执行的，而不是解析后的内联内容。因此，即使标签定义了等效的内联内容，以下内容也不匹配：

#### 示例 545

[酒吧][foo\!]

<p>[bar][foo!]</p>

[foo\!]: ./网址

链接标签 不能包含括号，除非使用反斜杠转义：

#### 示例 546

[foo][ref[]]

<p>[foo][ref[]]</p>  
<p>[参考[]: ./uri]</p>

[参考[]: ./uri]

#### 示例 547

[foo][ref[bar]]  
[ref[bar]]: • /uri

<p>[foo][ref[bar]]</p>  
<p>[ref[bar]]: • /uri</p>

#### 示例 548

[[[foo]]]  
[[[foo]]]: • /网址

<p>[[[foo]]]</p>  
<p>[[[foo]]]: • /网址</p>

#### 示例 549

[foo][ref[]]  
[参考\[]: • /uri

<p><a href="/uri">foo</a></p>

请注意，在此示例中，] 没有使用反斜杠转义：

#### 示例 550

[酒吧\\]: •/uri  
[酒吧\\]

<p><a href="/uri">酒吧\\</a></p>

一个链接标签 必须至少包含一个非空格、制表符或行尾的字符：

#### 示例 551

[]  
[]: /uri

<p>[]</p>  
<p>[]: • /uri</p>

#### 示例 552

[  
]  
[  
]: •/乌里

<p>[  
]</p>  
<p>[  
]: •/乌里</p>

一个折叠的参考链接 包括一个链接标签 那火柴 一个链接引用定义 文档中的其他位置，后跟字符串 []。链接标签的内容被解析为内联，用作链接的文本。链接的 URI 和标题由匹配的参考链接定义提供。因此，[foo][] 相当于[foo][foo].

#### 示例 553

[foo][]  
[foo]: •/网址 •“标题”

<p><a href="/网址" • title="title">foo</a></p>

#### 示例 554

[\*foo\* • 酒吧][]  
[\*foo\* • 酒吧]: •/网址 •“标题”

<p><a href="/网址" • title="title"><em>foo</em> •  
</p>

链接标签不区分大小写：

#### 示例 555

[美孚][]  
[foo]: •/网址 •“标题”

<p><a href="/网址" • title="title">Foo</a></p>

与完整参考链接一样，两组括号之间不允许有空格、制表符或行尾：

#### 示例 556

```
[foo] ·  
[]
```

```
<p><a href="/网址" · title="title">foo</a>  
[]</p>
```

[foo]: ·/网址 ·“标题”

一个**快捷方式参考链接** 包括一个**链接标签** 那**火柴** 一个**链接引用定义** 文档中的其他位置，并且后面没有 [] 或链接标签。链接标签的内容被解析为内联，用作链接的文本。链接的 URI 和标题由匹配的链接引用定义提供。因此，[foo] 相当于[foo][].

### 示例 557

```
[foo]
```

```
<p><a href="/网址" · title="title">foo</a></p>
```

[foo]: ·/网址 ·“标题”

### 示例 558

```
[*foo* · 酒吧]
```

```
<p><a href="/网址" · title="title"><em>foo</em></a> ·
```

[\*foo\* · 酒吧]: ./网址 ·“标题”

### 示例 559

```
[[*foo* · 酒吧]]
```

```
<p>[<一个 href="/网址" · title="title"><em>foo</em> ·
```

[\*foo\* · 酒吧]: ./网址 ·“标题”

### 示例 560

```
[[酒吧 · [foo]]
```

```
<p>[[酒吧 ·<— href="/url">foo</a>]</p>
```

[foo]: ·/网址

链接标签不区分大小写：

### 示例 561

```
[福]
```

```
<p><a href="/网址" · title="title">Foo</a></p>
```

[foo]: ·/网址 ·“标题”

链接文本后应保留一个空格：

### 示例 562

```
[foo] · 酒吧
```

```
<p><a href="/url">foo</a> · </p>
```

[foo]: ·/网址

如果您只想要括号内的文本，则可以使用反斜杠转义左括号以避免链接：

### 示例 563

```
\[foo]
```

```
<p>[foo]</p>
```

[foo]: ·/网址 ·“标题”

请注意，这是一个链接，因为链接标签以以下第一个右括号结尾：

### 示例 564

```
[foo*]: ·/网址
```

```
<p>*<a href="/url">foo*</a></p>
```

\* [foo\*]

完整和折叠的引用优先于快捷引用：

### 示例 565

[foo][酒吧]

<p><a href="/url2">foo</a></p>

[foo]: ./url1

[酒吧]: ./url2

## 示例 566

[foo][] <p><a href="/url1">foo</a></p>

[foo]: ./url1

内联链接同样优先：

## 示例 567

[foo]() <p><a href="">foo</a></p>

[foo]: ./url1

## 示例 568

[foo] (不是 ·—类联) <p><a href="/url1">foo</a> (不是 ·—链接) </p>

[foo]: ./url1

在以下情况下[酒吧][巴兹]被解析为引用，[foo]作为普通文本：

## 示例 569

[foo][bar][baz] <p>[foo]<a href="/url">酒吧</a></p>

[巴兹]: ./网址

不过，在这里，[foo][酒吧]被解析为引用，因为[酒吧]定义：

## 示例 570

[foo][bar][baz] <p><a href="/url2">foo</a><a href="/url1">巴兹</a></p>

[巴兹]: ./url1

[酒吧]: ./url2

这里 [foo] 不会被解析为快捷方式引用，因为它后面跟着一个链接标签（尽管 [酒吧] 未定义）：

## 示例 571

[foo][bar][baz] <p>[foo]<a href="/url1">酒吧</a></p>

[巴兹]: ./url1

[foo]: ./url2

## 6.4 图像

图像的语法类似于链接的语法，但有一点不同。[链接文本](#)，我们有一个[图片描述](#)。规则与[链接文本](#)，但(a) 图像描述以开头！[而不是 [，并且(b) 图像描述可能包含链接。图像描述具有内联元素作为其内容。当图像呈现为 HTML 时，这通常用作图像的替代属性。

## 示例 572

![foo](/url · “标题” ) <p><img alt="foo" title="标题" · </p>

## 示例 573

![foo · \* 酒吧\*] <p><img alt="foo · 酒吧" title="火车 · & 轨迹" · </p>

[foo · \* 酒吧\*]: ./火车.jpg · “火车 & 轨迹”

## 示例 574

![foo • !bar](/url)](/url2)

<p></p>

## 示例 575

![foo • [bar](/url)](/url2)

<p></p>

虽然本规范关注的是解析，而不是渲染，但建议在渲染 HTML 时，只渲染可以使用。请注意，在上面的例子中，alt 属性的值是 foo 酒吧，不是 foo [栏](/网址)或者 foo <a href="/url">酒吧</a>。仅呈现纯字符串内容，不带格式化。

## 示例 576

![foo • \* 酒吧\*][]

<p></p>

[foo •\* 酒吧\*]: ./火车.jpg • “火车 & 轨迹”

## 示例 577

![foo • \* 酒吧 \*][foobar]

<p></p>

[酒吧]: ./火车.jpg • “火车 & 轨迹”

## 示例 578

![foo](火车.jpg)

<p></p>

## 示例 579

我的[foo • 酒吧] (/路径/到/train.jpg • “标题” • • )

<p>我的 <图片src="/path/to/train.jpg" • alt="foo • 酒吧" • title="标题" •></p>

## 示例 580

![foo](<url>)

<p><img •源文件="url" • alt="foo" •></p>

## 示例 581



<p><img •源文件= “/网址” • 替代= “” •></p>

参考文献格式:

## 示例 582

![foo][bar]

<p><img •源文件= “/网址” • alt="foo" •></p>

[酒吧]: ./网址

## 示例 583

![foo][bar]

<p><img •源文件= “/网址” • alt="foo" •></p>

[酒吧]: ./网址

折叠:

## 示例 584

![foo][]

<p><img •源文件= “/网址” • alt="foo" • title="标题" •></p>

[foo]: ./网址 •“标题”

## 示例 585

![\*foo\* • 酒吧][]

<p><img •源文件= “/网址” • alt="foo • 酒吧" title="标题" • /></p>

[\*foo\* • 酒吧]: ./网址 •“标题”

标签不区分大小写:

## 示例 586

```
![Foo]()
```

```
<p><img alt="Foo" title="标题" />
```

```
[foo]: ./网址 "标题"
```

与参考链接一样，两组括号之间不允许有空格、制表符和行尾：

#### 示例 587

```
![foo] .  
[]
```

```
[foo]: ./网址 "标题"
```

捷径：

#### 示例 588

```
![foo]
```

```
<p><img alt="foo" title="标题" />
```

```
[foo]: ./网址 "标题"
```

#### 示例 589

```
![*foo* · 酒吧]
```

```
<p><img alt="foo · 酒吧" title="标题" />
```

```
[*foo* · 酒吧]: ./网址 "标题"
```

请注意，链接标签不能包含未转义的括号：

#### 示例 590

```
![[foo]]
```

```
<p>![[foo]]</p>
```

```
[[foo]]: ./网址 "标题"
```

链接标签不区分大小写：

#### 示例 591

```
![Foo]
```

```
<p><img alt="Foo" title="标题" />
```

```
[foo]: ./网址 "标题"
```

如果你只想要一个文字！后面跟着括号内的文本，你可以使用反斜杠转义开头的 [:

#### 示例 592

```
!\[foo]
```

```
<p>!\[foo]</p>
```

```
[foo]: ./网址 "标题"
```

如果要在文字！后添加链接，请用反斜杠转义！:

#### 示例 593

```
\![foo]
```

```
<p>!<a href="/网址" title="title">foo</a></p>
```

```
[foo]: ./网址 "标题"
```

## 6.5 自动链接

[自动链接](#) < 和 > 内的 s 是绝对 URI 和电子邮件地址。它们被解析为链接，以 URL 或电子邮件地址作为链接标签。

一个[URI 自动链接](#)由 < 组成，后跟[绝对 URI](#) 后跟 >。它被解析为指向 URI 的链接，并以 URI 作为链接的标签。

一个[绝对 URI](#)，为此目的，由一个[方案](#)后跟冒号 (:) 后跟零个或多个除以下字符之外的字符[ASCII 控制字符](#)，[空间](#)、<和>。如果 URI 包含这些字符，则必须对其进行百分比编码（例如 %20 表示空格）。

就本规范而言，[方案](#)是以 ASCII 字母开头的 2 到 32 个字符的任意序列，后跟 ASCII 字母、数字或符号加号（“+”）、句点（“.”）或连字符（“-”）。

以下是一些有效的自动链接：

#### [示例 594](#)

```
<http://foo.bar.baz> <p><a .  
 href="http://foo.bar.baz">http://foo.bar.baz</a></p>
```

#### [示例 595](#)

```
<https://foo.bar.baz/test?q=hello&id=22&boolean> <p><a .  
 href="https://foo.bar.baz/test?  
 q=hello&id=22&boolean">https://foo.bar.baz /test?  
 q=hello&id=22&boolean</a></p>
```

#### [示例 596](#)

```
<irc://foo.bar:2233/baz> <p><a .  
 href="irc://foo.bar:2233/baz">irc://foo.bar:2233/baz</  
 a></p>
```

大写也可以：

#### [示例 597](#)

```
<邮箱：FOO@BAR.BAZ > <p><a .  
 href="邮箱：FOO@BAR.BAZ ">邮箱：FOO@BAR.BAZ </a></p>
```

请注意，许多被视为[绝对 URI](#)对于本规范的目的而言，不是有效的 URI，因为它们的方案未注册或其语法存在其他问题：

#### [示例 598](#)

```
<a+b+c:d> <p><a .  
 href="a+b+c:d">a+b+c:d</a></p>
```

#### [示例 599](#)

```
<虚构方案://foo,bar> <p><a .  
 href="made-up-scheme://foo,bar">made-  
 upscheme://foo,bar</a></p>
```

#### [示例 600](#)

```
<https://../> <p><a .  
 href="https://../">https://..</a></p>
```

#### [示例 601](#)

```
<本地主机： 5001/foo> <p><a .  
 href="localhost:5001/foo">本地主机： 5001/foo</a></  
 p>
```

自动链接中不允许有空格：

#### [示例 602](#)

```
<https://foo.bar/baz · 比姆> <p><https://foo.bar/baz · .</p>
```

反斜杠转义在自动链接中不起作用：

#### [示例 603](#)

```
<https://example.com/\[\]> <p><a .  
 href="https://example.com/%5C%5B%5C">https://  
 example.com/\[\]</a></p>
```

一个**电子邮件自动链接**由<组成，后跟**电子邮件**，后跟>。链接的标签是电子邮件地址，URL是邮箱：然后是电子邮件地址。

一个**电子邮件**，为了这些目的，是任何符合**HTML5 规范中的非规范正则表达式**：

```
/^@[a-zA-Z0-9.!#$%&!*+/?^_`{|}~-]+@[a-zA-Z0-9](?:[a-zA-Z0-9-]{0,61}[a-zA-Z0-9])?(?:\.[a-zA-Z0-9](?:[a-zA-Z0-9-]{0,61}[a-zA-Z0-9])?)*$/
```

电子邮件自动链接的示例：

#### 示例 604

```
<foo@bar.example.com> <p><a href="mailto:foo@bar.example.com">foo@bar.example.com</a></p>
```

#### 示例 605

```
<foo+special@Bar.baz-bar0.com> <p><a href="mailto:foo+special@Bar.bazbar0.com">foo+special@Bar.baz-bar0.com</a></p>
```

反斜杠转义在电子邮件自动链接中不起作用：

#### 示例 606

```
<foo\ +@bar.example.com> <p><foo+@bar.example.com></p>
```

这些不是自动链接：

#### 示例 607

```
<> <p><></p>
```

#### 示例 608

```
< https://foo.bar . > <p><. https://foo.bar .></p>
```

#### 示例 609

```
<m:abc> <p><m:abc></p>
```

#### 示例 610

```
<foo.bar.baz> <p><foo.bar.baz></p>
```

#### 示例 611

```
https://example.com <p>https://example.com</p>
```

#### 示例 612

```
foo@bar.example.com <p>foo@bar.example.com</p>
```

## 6.6 原始 HTML

<和>之间的文本看起来像 HTML 标记，将被解析为原始 HTML 标记，并将以 HTML 格式呈现，无需转义。标记和属性名称不限于当前 HTML 标记，因此可以使用自定义标记（甚至 DocBook 标记）。

标签的语法如下：

一个**标签名称**由一个 ASCII 字母和零个或多个 ASCII 字母、数字或连字符 (-) 组成。

一个**属性**由空格、制表符和最多一行结尾组成，**属性名称**以及可选的**属性值规范**。

一个**属性名称**由一个 ASCII 字母、\_ 或 : 组成，后面跟着零个或多个 ASCII 字母、数字、\_、.、: 或 -。 （注意：这是仅限于 ASCII 的 XML 规范。HTML5 比较宽松。）

一个[属性值规范](#)由可选空格、制表符和最多一个行尾字符、一个 = 字符、可选空格、制表符和最多一个行尾字符以及一个[属性值](#)。

一个[属性值](#)包括一个[不带引号的属性值](#)，一个[单引号属性值](#)或[双引号属性值](#)。

一个[不带引号的属性值](#)是一个非空字符串，不包括空格、制表符、行尾、“、'、=、<、>或`。

一个[单引号属性值](#)由'、零个或多个字符（不包括'）和最后的'组成。

一个[双引号属性值](#)由“、零个或多个字符（不包括“）以及最后的“组成。

一个[打开标签](#)由一个<字符、一个[标签名称](#)，零个或多个[属性](#)、可选的空格、制表符、最多一个行尾、可选的 / 字符和>字符。

一个[结束标签](#)由字符串</、[标签名称](#)、可选的空格、制表符、最多一个行尾以及字符>。

一个[HTML注释](#)由<!-->、<!---->或<!--（不包括字符串-->和-->的字符串）组成（请参阅[HTML规范](#)）。

一个[加工指令](#)由字符串<?、不包括字符串?>的字符串和字符串?>组成。

一个[宣言](#)由字符串<!、ASCII字母、零个或多个字符（不包括字符）组成  
>以及字符>。

一个[CDATA部分](#)由字符串<![CDATA[，不包括字符串]]>和字符串]]>的字符串。

一个[HTML标签](#)包括一个[打开标签](#)，一个[结束标签](#)，一个[HTML注释](#)，一个[加工指令](#)，一个[宣言](#)或[CDATA部分](#)。

以下是一些简单的开放标签：

#### [示例 613](#)

<a><bab><c2c>

<p><a><bab><c2c></p>

空元素：

#### [示例 614](#)

<a/><b2/>

<p><a/><b2/></p>

允许使用空格：

#### [示例 615](#)

<— <b2> /  
数据= “foo” · >

<p><a · ·<b2> /  
数据= “foo” · ></p>

具有属性：

#### [示例 616](#)

<— foo=“酒吧” ·bam=·巴茲 ·em> “</em>”  
\_boolean ·变焦：33=变焦：33 · />

<p><a foo=“酒吧” ·bam=·巴茲 ·em> “</em>”  
\_boolean ·变焦：33=变焦：33 · </p>

可以使用自定义标签名称：

#### [示例 617](#)

福 · <响应式图像 · src="foo.jpg" · />

<p>Foo · <响应式图像 · src="foo.jpg" · </p>

非法标签名称，无法解析为 HTML：

#### [示例 618](#)

非法的属性名称：

### 示例 619

```
<—h*#ref="hi">
```

```
<p><a · h*#ref="hi"></p>
```

非法属性值：

### 示例 620

```
<—href="嗨'> · <—href=嗨'>
```

```
<p><a · href="嗨'> · — · href=hi'></p>
```

非法空格：

### 示例 621

```
< <—><
<酒吧/ ·   >
<foo `酒吧=巴兹
比姆！波普 ./>
```

```
<p>< · 一个><
foo><bar/ ·       >
<foo · 酒吧=巴兹
比姆！波普 </p>
```

缺少空格：

### 示例 622

```
<—href='bar'title=标题>
```

```
<p><a · href='bar'title=title></p>
```

结束标签：

### 示例 623

```
</a></foo ·>
```

```
<p><a></foo> · ></p>
```

结束标记中的非法属性：

### 示例 624

```
</—href="foo">
```

```
<p></a> · href="foo"></p>
```

评论：

### 示例 625

```
富 · <!-- · 这 · 是 一 个 ·
评论 · - 和 · 连字符 ·-->
```

```
<p>foo ·<!-- · 这 · 是 一 个 ·
评论 · - 和 · 连字符 ·--></p>
```

### 示例 626

```
富 · <!--> · 富 · -->
```

```
<p>foo ·<!--> · 富 · --></p>
<p>foo ·<!--> · 富 · --></p>
```

```
富 · <!--> · 富 · -->
```

处理说明：

### 示例 627

```
富 · <?php · 回声 $—>; >
```

```
<p>foo ·<?php · 回声 $—>; ? ></p>
```

声明：

### 示例 628

```
富 · <!元素 · br 空>
```

```
<p>foo ·<!元素 · br 空的></p>
```

CData 部分：

### 示例 629

富 · <![CDATA[>&<]]>

```
<p>foo ·<![CDATA[>&<]]></p>
```

实体和数字字符引用保留在 HTML 属性中：

### 示例 630

富 · <—href="ö">

```
<p>foo ·<—href="ö"></p>
```

反斜杠转义在 HTML 属性中不起作用：

### 示例 631

富 · <—href="\"\*>

```
<p>foo ·<—href="\"*></p>
```

### 示例 632

<—href="\"">

```
<p><a · href=""></p>
```

## 6.7 硬换行

如果行尾（不在代码段或 HTML 标记中）以两个或多个空格开头，且不在块的末尾，则将被解析为**硬换行**（在 HTML 中呈现为<br />标签）：

### 示例 633

富 · ·  
巴兹

```
<p>foo<br ·>  
</p>
```

为了更明显，在行结束符可以代替两个或多个空格：

### 示例 634

foo\  
巴兹

```
<p>foo<br ·>  
</p>
```

可以使用两个以上的空格：

### 示例 635

富 · · · · ·  
巴兹

```
<p>foo<br ·>  
</p>
```

下一行开头的前导空格将被忽略：

### 示例 636

富 · ·  
· · 酒吧

```
<p>foo<br ·>  
</p>
```

### 示例 637

foo\  
· · 酒吧

```
<p>foo<br ·>  
</p>
```

硬换行可以出现在强调、链接和其他允许内联内容的结构内：

### 示例 638

\* foo · ·  
酒吧\*

```
<p><em>foo</br> ·>  
</em></p>
```

### 示例 639

```
* foo\
酒吧*
```

<p><em>foo</br> .</em></p>

代码跨度内不会出现硬换行

#### 示例 640

```
`代码 . .
跨度`
```

<p><code>代码 . . </p> <p>跨度</p>

#### 示例 641

```
`代码 \
跨度`
```

<p><code>代码\ . </p> <p>跨度</p>

或 HTML 标签：

#### 示例 642

```
<--href="foo . .
酒吧 ">
```

<p><a href="foo . .
酒吧" ></p>

#### 示例 643

```
<--href="foo\
酒吧 ">
```

<p><a href="foo\
酒吧" ></p>

硬换行符用于分隔块内的内联内容。硬换行符的语法在段落或其他块元素的末尾均不起作用：

#### 示例 644

```
foo\

```

<p>foo</p>

#### 示例 645

```
富 . .

```

<p>foo</p>

#### 示例 646

```
### foo\

```

<h3>foo</h3>

#### 示例 647

```
### 富 . .

```

<h3>foo</h3>

## 6.8 软换行

常规行尾（不在代码段或 HTML 标记中）如果前面没有两个或多个空格或反斜杠，则将被解析为软中断。（软换行符在 HTML 中可以呈现为行结束符或空格。在浏览器中结果相同。在此处的示例中，行结束符将被使用。）

#### 示例 648

```
富
巴兹
```

<p>foo
</p>

删除行末和下一行开头的空格：

#### 示例 649

```
富 .
巴兹
```

<p>foo
</p>

符合要求的解析器 可能会将 HTML 中的软换行符呈现为行尾或空格。

渲染器还可以提供将软换行符渲染为硬换行符的选项。

## 6.9 文本内容

任何未根据上述规则进行解释的字符都将被解析为纯文本内容。

### 示例 650

你好。\$.;'那里

<p>你好。\$.;'那里</p>

### 示例 651

福。xpñv

<p>Foo .</p>

内部空间被逐字保留：

### 示例 652

多种的· · · · 空格

<p>多个· · · · 空格</p>

---

## 附录：解析策略

在本附录中，我们描述了 CommonMark 参考实现中使用的解析策略的一些特点。

### 概述

解析分为两个阶段：

1. 在第一阶段，输入行被使用，文档的块结构（将其划分为段落、块引用、列表项等）被构建。文本被分配给这些块，但不会被解析。链接引用定义被解析，并构建链接映射。
2. 第二阶段，利用第一阶段构建的链接引用图，将段落和标题的原始文本内容解析为 Markdown 内联元素序列（字符串、代码跨度、链接、强调等）。

在处理的每个阶段，文档都表示为**块**。树的根是文档块。文档可能有任何数量的其他块**孩子们**。这些子块又可以有其他块作为子块。块的最后一个子块通常被认为是**打开**，这意味着后续的输入行可以改变其内容。（未打开的块是**关闭**）例如，这是一个可能的文档树，其中打开的块用箭头标记：

```
-->文档
-->block_quote
    段落
        “Lorem ipsum dolor\nsit amet。”
    ->列表 (type=bullet tight=true bullet_char=-)
        list_item
            段落
                “谁是伊拉克人”
        ->list_item
            ->段落
                “aliquando id”
```

### 第一阶段：区块结构

处理的每一行都会影响此树。将分析每一行，并根据其内容，以下列一种或多种方式更改文档：

1. 一个或多个打开的区块可能被关闭。
2. 可以创建一个或多个新块作为最后一个打开的块的子块。
3. 可以将文本添加到树上剩余的最后一个（最深的）开放块中。

一旦某行以这种方式被合并到树中，它就可以被丢弃，因此可以以流的形式读取输入。

对于每一行，我们都遵循以下步骤：

1. 首先，我们遍历打开的块，从根文档开始，然后向下遍历最后一个子块，直到最后一个打开的块。每个块都规定了一个条件，如果块要保持打开状态，该行必须满足该条件。例如，块引用需要一个>字符。段落需要一个

非空白行。在此阶段，我们可能会匹配所有或部分打开的块。但我们还不能关闭未匹配的块，因为我们可能有一个惰性延续行。

2. 接下来，在消耗完现有区块的延续标记后，我们寻找新的区块开始（例如  
-> 用于块引用）。如果遇到新的块开始，我们会关闭步骤 1 中未匹配的所有块，然后将新块创建为最后一个匹配的容器块的子块。
3. 最后，我们查看行的剩余部分（在块标记（如 >、列表标记和缩进）被使用之后）。这是可以合并到最后一个开放块（段落、代码块、标题或原始 HTML）中的文本。

当我们看到段落中的一行时，就会形成 Setext 标题setext 标题下划线。

当段落关闭时，会检测引用链接定义；解析累积的文本行以查看它们是否以一个或多个引用链接定义开头。任何剩余部分都会变成普通段落。

通过考虑上面的树是如何由四行 Markdown 生成的，我们可以看到这是如何工作的：

```
> Lorem ipsum dolor
sit amet。
> — 伊拉克利翁的习俗
> - aliquando id
```

一开始，我们的文档模型只是

-->文档

我们文本的第一行，

```
> Lorem ipsum 悲伤
```

导致区块引用块被创建为我们开放的子块文档块，以及段落块作为block\_quote。然后将文本添加到最后一个打开的块，段落：

```
-->文档
-->block_quote
-> 段落
    “Lorem ipsum 悲痛”
```

下一行，

坐下吧。

是开放的“惰性延续”段落，因此它被添加到段落的文本中：

```
-->文档
-->block_quote
-> 段落
    “Lorem ipsum dolor\nsit amet。”
```

第三行，

```
> — 伊拉克利翁的习俗
```

导致段落块被关闭，并一个新的列表作为子块打开的block\_quote。一个列表项还被添加为列表，和一个段落作为列表项。然后将文本添加到新的段落：

```
-->文档
-->block_quote
-> 段落
    “Lorem ipsum dolor\nsit amet。”
-> 列表 (type=bullet tight=true bullet_char=-)
-> list_item
-> 段落
    “谁是伊拉克人”
```

第四行，

```
> - aliquando id
```

导致列表项（和它的孩子段落）将被关闭，并列表项作为孩子开放列表。一个段落作为新子项添加列表项，包含文本。这样我们就得到了最终的树：

```
-->文档
-->block_quote
    段落
        "Lorem ipsum dolor\nsit amet."
->列表 (type=bullet tight=true bullet_char=-)
list_item
    段落
        "谁是伊拉克人"
->list_item
    -> 段落
        "aliquando id"
```

## 第二阶段：内联结构

一旦解析了所有输入，所有打开的块都会被关闭。

然后我们“遍历树”，访问每个节点，并将段落和标题的原始字符串内容解析为内联。此时，我们已经看到了所有链接引用定义，因此我们可以在过程中解析引用链接。

```
文档
区块引用
段落
    str "Lorem ipsum
dolor" 软中断
    str "坐下吧。"
列表 (类型=bullet tight=true bullet_char=-)
列表项
    段落
        斯特 "Qui"
        强调
            str "quodsi iracundia"
    列表项
    段落
        str "aliquando id"
```

注意行结束符 第一段被解析为软中断，第一个列表项中的星号已变为强调。

## 嵌套强调和链接解析算法

到目前为止，内联解析中最棘手的部分是处理强调、强调、链接和图像。这是使用以下算法完成的。

当我们解析内联时，我们遇到了

- 一连串 \* 或 \_ 字符，或者 [ 或 !
- [

我们插入一个以这些符号为文字内容的文本节点，并将指向该文本节点的指针添加到[分隔符堆栈](#)。

这[分隔符堆栈](#)是一个双向链表。每个元素包含一个指向文本节点的指针，以及有关

- 分隔符的类型 ([、![]、\*、\_]) 分
- 隔符的数量，
- 分隔符是否处于“活动”状态（所有分隔符都处于活动状态），以及
- 分隔符是否是潜在的开启符、潜在的结束符、或两者兼有（这取决于分隔符前后的字符类型）。

当我们点击 ] 字符时，我们调用[寻找链接或图片程序](#)（见下文）。

当我们到达输入末尾时，我们调用[过程强调程序](#)（见下文），堆栈底部=无效的。

## 寻找链接或图片

我们从分隔符堆栈的顶部开始，向后查找堆栈中的开头的 [ 或 ![ 分隔符。

- 如果找不到，我们将返回文字文本节点]。
- 如果我们确实找到了一个，但它不是积极的，我们从堆栈中删除非活动分隔符，并返回文字文本节点]。
- 如果我们找到一个并且它处于活动状态，那么我们就会提前解析以查看是否有内联链接/图像，参考链接/图像，折叠参考链接/图像或快捷参考链接/图像。
  - 如果没有，则我们从分隔符堆栈中删除开头分隔符并返回文字文本节点]。
  - 如果我们这样做，那么
    - 我们返回一个链接或图像节点，其子节点是开启分隔符指向的文本节点后的内联节点。
    - 我们跑过程强调在这些内联中，使用 [ 开启符作为堆栈底部。
    - 我们删除开头的分隔符。
    - 如果我们有一个链接（而不是图像），我们还将打开分隔符之前的所有 [ 分隔符设置为不活跃。（这将阻止我们获取链接内的链接。）

## 过程强调

范围堆栈底部设定了我们下降到什么程度的下限分隔符堆栈。如果它是 NULL，我们可以一直走到底部。否则，我们在访问之前停止堆栈底部。

让当前位置指向元素分隔符堆栈就在上方堆栈底部（或第一个元素，如果堆栈底部为 NULL）。

我们跟踪openers\_bottom对于每种分隔符类型 (\*, \_)，索引到结束分隔符运行的长度（模 3）以及结束分隔符是否也可以是开启分隔符。将其初始化为 堆栈底部。

然后我们重复以下步骤，直到没有可能的终结者：

- 移动当前位置在分隔符堆栈中向前移动（如果需要），直到我们找到第一个带有分隔符 \* 或 \_ 的潜在结束符。（这将是最接近输入开头的潜在结束符 - 按解析顺序的第一个。）
- 现在，回顾一下堆栈（保持在上方堆栈底部和openers\_bottom对于这种分隔符类型），第一个匹配的潜在开启符（“匹配”表示相同的分隔符）。
- 如果找到：
  - 弄清楚我们是否有强调或强烈强调：如果接近和开放跨度的长度都  $\geq 2$ ，则我们有强烈的强调，否则就是常规的。
  - 在与开头对应的文本节点后，相应地插入一个强调或强强强节点。
  - 从分隔符堆栈中删除开启符和结束符之间的所有分隔符。
  - 从开始和结束文本节点中删除 1 个（常规强调）或 2 个（强强调）分隔符。如果结果为空，则删除它们并删除分隔符堆栈的相应元素。如果结束节点被删除，则重置当前位置到堆栈中的下一个元素。
- 如果没有找到：
  - 放openers\_bottom到之前的元素当前位置。（我们知道，到目前为止，还没有找到这种更接近这一点的开场白，所以这为未来的搜索设置了一个下限。）
  - 如果接近当前位置不是潜在的开启器，将其从分隔符堆栈中移除（因为我们知道它也不可能关闭器）。
  - 进步当前位置到堆栈中的下一个元素。

完成后，我们删除上面的所有分隔符堆栈底部来自分隔符堆栈。