

# Markdown入门之基础篇



## Markdown基础

---

[基础篇视频讲解链接](#) [画图篇视频讲解链接](#)

### 标题

---

# 标题名字（井号的个数代表标题的级数）

## 一级标题使用1个#

---

## 二级标题使用2个#

---

### 三级标题使用3个#

#### 四级标题使用4个#

#### 五级标题使用5个#

#### 六级标题使用6个#

##### 最多支持六级标题#

# 文字

## 删除线

这就是 ~~删除线~~ (使用波浪号)

这就是 ~~删除线~~ (使用波浪号)

## 斜体

这是用来 *\*斜体\** 的 文本

这是用来 *斜体* 的 文本

## 加粗

这是用来 **\*\*加粗\*\*** 的 文本

这是用来 **加粗** 的 文本

## 斜体+加粗

这是用来 ***\*\*\*斜体+加粗\*\*\**** 的 文本

这是用来 ***斜体+加粗*** 的 文本

## 下划线

下划线是HTML语法

下划线 下划线(快捷键 `command + u`, 视频中所有的快捷键都是针对Mac系统, 其他系统可自行查找)

## 高亮（需勾选扩展语法）

这是用来 ***==斜体+加粗==*** 的文本

这是用来 ***斜体+加粗*** 的文本

## 下标（需勾选扩展语法）

水 H~2~O  
双氧水 H~2~O~2~

水 H<sub>2</sub>O


双氧水  $\text{H}_2\text{O}_2$

## 上标（需勾选扩展语法）

```
面积 m^2^
体积 m^3^
```

面积  $\text{m}^2$  体积  $\text{m}^3$

## 表情符号

Emoji 支持表情符号，你可以用系统默认的 Emoji 符号（Windows 用户不一定支持，自己试下~）。也可以用图片的表情，输入  将会出现智能提示。

### 一些表情例子

```
:smile: :laughing: :dizzy_face: :sob: :cold_sweat: :sweat_smile: :cry:
:triumph: :heart_eyes: :relaxed: :sunglasses: :weary:

:+1: :-1: :100: :clap: :bell: :gift: :question: :bomb: :heart: :coffee:
:cyclone: :bow: :kiss: :pray: :sweat_drops: :hankey: :exclamation: :anger:
```



( Mac: `control + command + space` 点选)

## 表格

使用 `|` 来分隔不同的单元格，使用 `-` 来分隔表头和其他行：

```
name | price
--- | ---
fried chicken | 19
cola|5
```

为了使 Markdown 更清晰，`|` 和 `-` 两侧需要至少有一个空格（最左侧和最右侧的 `|` 外就不需要了）。

name	price
fried chicken	19
cola	5

为了美观，可以使用空格对齐不同行的单元格，并在左右两侧都使用 `|` 来标记单元格边界，在表头下方的分隔线标记中加入 `:`，即可标记下方单元格内容的对齐方式：

	name		price	
	:-----		:---	
	fried chicken		19	
	cola		32	

name	price
fried chicken	19
cola	32

使用快捷键 `command + opt + T` 更方便(段落→表格→插入表格，即可查看快捷键)

# 引用

>“后悔创业”

“后悔创业”

>也可以在引用中  
>>使用嵌套的引用

也可以在引用中  
使用嵌套的引用

# 列表

## 无序列表--符号 空格

\* 可以使用 `\*` 作为标记  
+ 也可以使用 `+`  
- 或者 `-`

- 可以使用 `\*` 作为标记
- 也可以使用 `+`
- 或者 `-`

## 有序列表--数字 . 空格

1. 有序列表以数字和 `1.` 开始；
3. 数字的序列并不会影响生成的列表序列；
4. 但仍然推荐按照自然顺序（1.2.3...）编写。

1. 有序列表以数字和 `1.` 开始；
2. 数字的序列并不会影响生成的列表序列；
3. 但仍然推荐按照自然顺序（1.2.3...）编写。

可以使用：数字\。来取消显示为列表（用反斜杠进行转义）

## 代码

### 代码块

````语言名称`

```
public static void main(String[] args) {  
    }
```

### 行内代码

也可以通过 ```，插入行内代码（``` 是 ``Tab`` 键上边、数字 ``1`` 键左侧的那个按键）：

例如 ``Markdown``

Markdown

### 转换规则

代码块中的文本（包括 Markdown 语法）都会显示为原始内容

### 分隔线

可以在一行中使用三个或更多的 `*`、`-` 或 `_` 来添加分隔线（```）：

```
***  
----  
_
```

# 跳转

## 外部跳转--超链接

格式为 `[link text](link)`。

[\[帮助文档\]](https://support.typora.io/Links/#faq)(<https://support.typora.io/Links/#faq>)

[帮助文档](#)

## 内部跳转--本文件内跳（Typora支持）

格式为 `[link text](#要去的目的地--标题)`。

[\[我想跳转\]](#)([#饼图 \(Pie\)](#) )

Open Links in Typora

You can use `command+click` (macOS), or `ctrl+click` (Linux/Windows) on links in Typora to jump to target headings, or open them in Typora, or open in related apps.

[我想跳转](#)

## 自动链接

使用 `<>` 包括的 URL 或邮箱地址会被自动转换为超链接：

[<https://www.baidu.com>](https://www.baidu.com)

[<123@email.com>](mailto:123@email.com)

<https://www.baidu.com>

[123@email.com](mailto:123@email.com)

## 图片

![\[自己起的图片名字\]](#) (图片地址或者图片本地存储的路径)

## 网上的图片

![\[friedChicken\]](#)([https://ss0.bdstatic.com/94oJfD\\_bAaCT8t7mm9GUKT-xh\\_/timg?image&quality=100&size=b4000\\_4000&sec=1580814517&di=2630beac440e5dab0e44c7286a3b2b61&src=http://imgsrc.baidu.com/forum/w=580/sign=12c730c4ff03738dde4a0c2a831ab073/9497794f9258d1091818e6d6d858ccbf6d814d1b.jpg](https://ss0.bdstatic.com/94oJfD_bAaCT8t7mm9GUKT-xh_/timg?image&quality=100&size=b4000_4000&sec=1580814517&di=2630beac440e5dab0e44c7286a3b2b61&src=http://imgsrc.baidu.com/forum/w=580/sign=12c730c4ff03738dde4a0c2a831ab073/9497794f9258d1091818e6d6d858ccbf6d814d1b.jpg))



## 本地图片

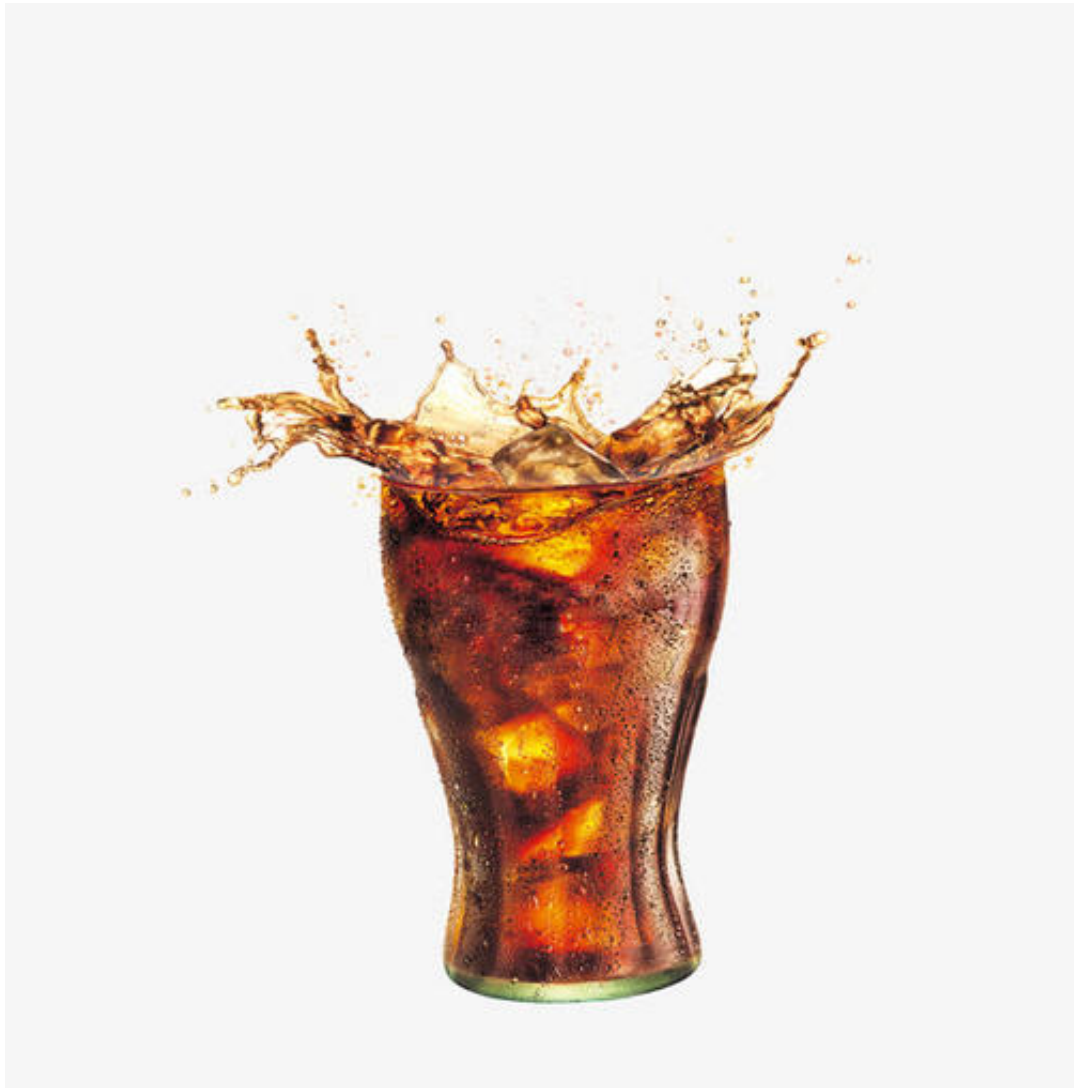
```
![friedChicken](friedChicken.jpg)
```

在同一个文件夹里（用相对路径）

或者直接拷贝







利用Markdown画图（需勾选扩展语法）

---

# Markdown入门之画图篇



救救

markdown画图也是轻量级的，功能并不全。

Mermaid 是一个用于画流程图、状态图、时序图、甘特图的库，使用 JS 进行本地渲染，广泛集成于许多 Markdown 编辑器中。Mermaid 作为一个使用 JS 渲染的库，生成的不是一个“图片”，而是一段 HTML 代码。

(不同的编辑器渲染的可能不一样)

## 流程图(graph)

### 概述

graph 方向描述

[图表中的其他语句...](#)

关键字graph表示一个流程图的开始，同时需要指定该图的方向。

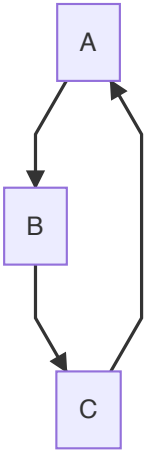
其中“方向描述”为：

用词	含义
TB	从上到下
BT	从下到上
RL	从右到左
LR	从左到右

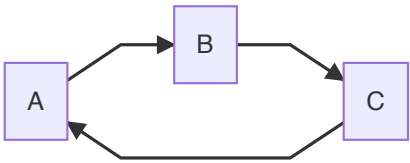
T = TOP, B = BOTTOM, L = LEFT, R = RIGHT, D = DOWN

最常用的布局方向是TB、LR。

```
graph TB;
  A-->B
  B-->C
  C-->A
```



```
graph LR;
  A-->B
  B-->C
  C-->A
```



流程图常用符号及含义

节点形状

表述	说明	含义
id[文字]	矩形节点	表示过程，也就是整个流程中的一个环节
id(文字)	圆角矩形节点	表示开始和结束
id((文字))	圆形节点	表示连接。为避免流程过长或有交叉，可将流程切开。成对
id{文字}	菱形节点	表示判断、决策
id>文字]	右向旗帜状节点	

**单向箭头线段：**表示流程进行方向

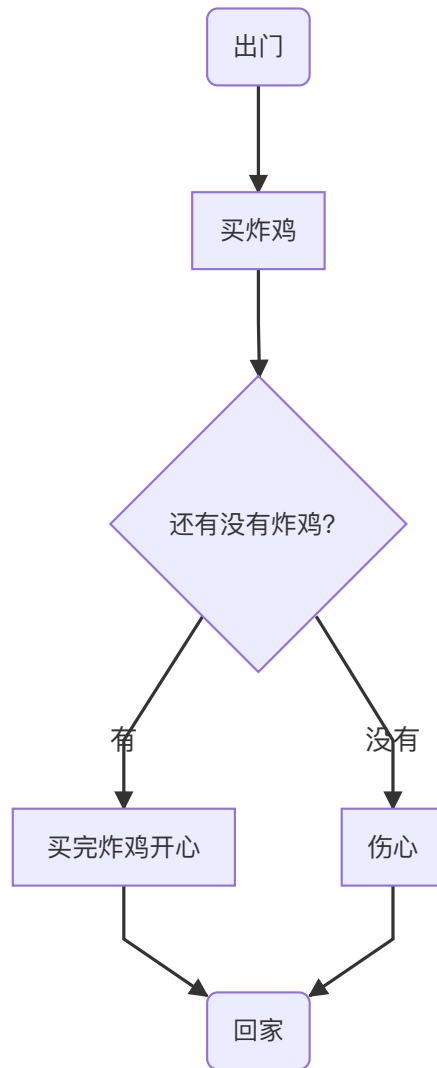
id即为节点的唯一标识，A~F 是当前节点名字，类似于变量名，画图时便于引用

括号内是节点中要显示的文字，默认节点的名字和显示的文字都为A

```
graph TB
  A
  B(圆角矩形节点)
  C[矩形节点]
  D((圆形节点))
  E{菱形节点}
  F>右向旗帜状节点]
```

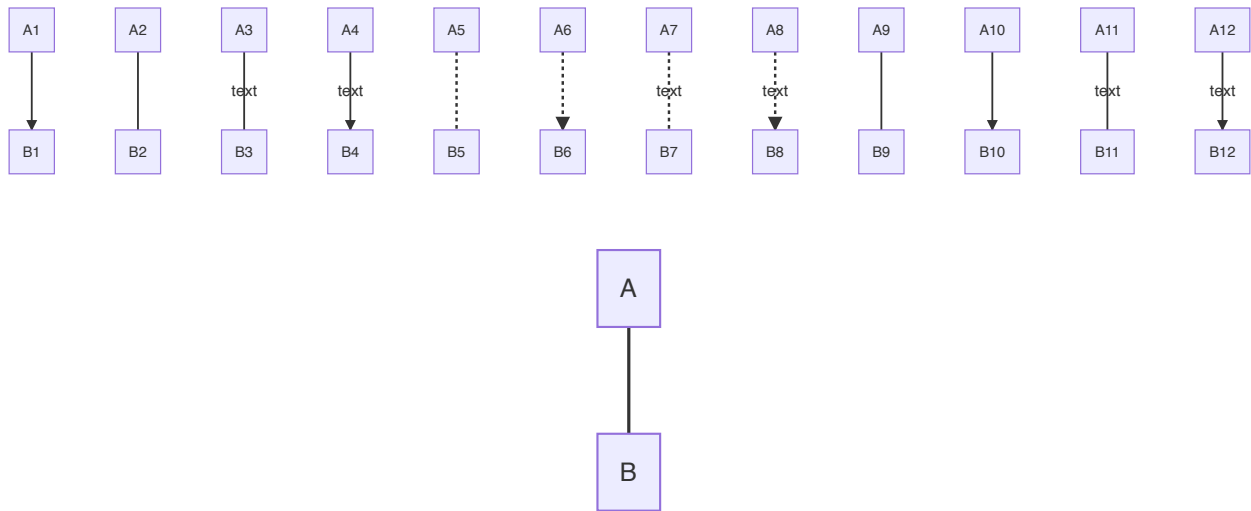


```
graph TB
  begin(出门)--> buy[买炸鸡]
  buy --> IsRemaining{"还有没有炸鸡?"}
  IsRemaining -->|有| happy[买完炸鸡开心]--> goBack(回家)
  IsRemaining -->|没有| sad["伤心"]--> goBack
```



## 连线

```
graph TB
  A1-->B1
  A2---B2
  A3--text---B3
  A4--text-->B4
  A5-.-B5
  A6-.->B6
  A7-.text.-B7
  A8-.text.->B8
  A9===B9
  A10==>B10
  A11==text===B11
  A12==text==>B12
```

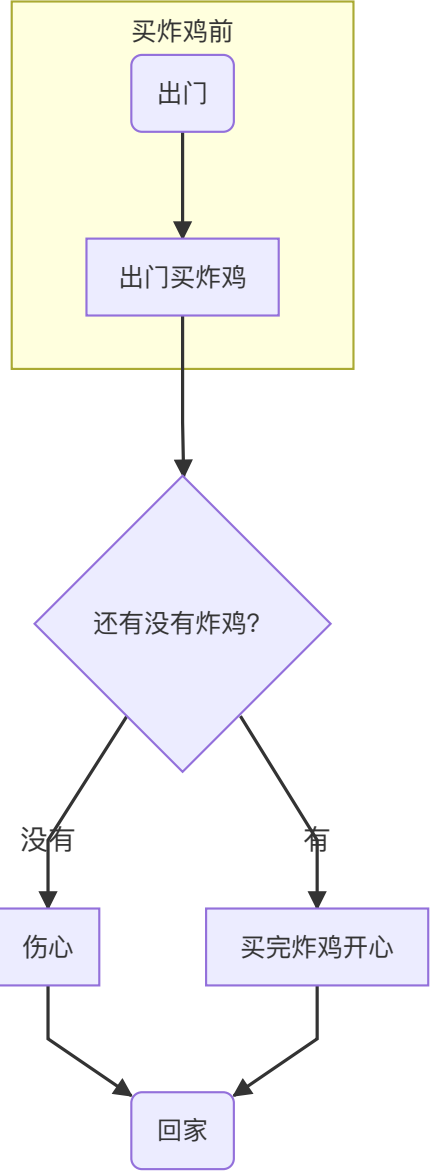


## 子图表

使用以下语法添加子图表

```
subgraph 子图表名称
    子图表中的描述语句...
end
```

```
graph TB
    subgraph 买炸鸡前
        begin(出门)--> buy[出门买炸鸡]
    end
    buy --> IsRemaining{"还有没有炸鸡? "}
    IsRemaining --没有--> sad["伤心"]--> goBack(回家)
    IsRemaining -->|有| happy[买完炸鸡开心]--> goBack
```



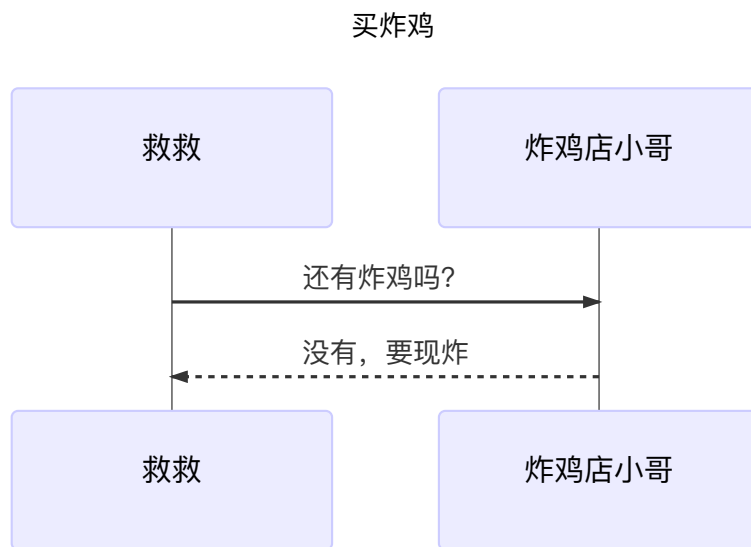
## 序列图(sequence diagram)

### 概述

```
sequenceDiagram
    [参与者1][消息线][参与者2]:消息体
    ...
```

`sequenceDiagram` 为每幅时序图的固定开头

```
sequenceDiagram
    Title: 买炸鸡
    救救->>炸鸡店小哥: 还有炸鸡吗?
    炸鸡店小哥-->>救救: 没有, 要现炸
```



## 参与者 (participant)

传统时序图概念中参与者有角色和类对象之分，但这里我们不做此区分，用参与者表示一切参与交互的事物，可以是人、类对象、系统等形式。中间竖直的线段从上至下表示时间的流逝。

```
sequenceDiagram
    participant 参与者 1
    participant 参与者 2
    ...
    participant 简称 as 参与者 3 #该语法可以在接下来的描述中使用简称来代替参与者 3
```

`participant <参与者名称>` 声明参与者，语句次序即为参与者横向排列次序。

## 消息线

类型	描述
->	无箭头的实线
-->	无箭头的虚线
->>	有箭头的实线（主动发出消息）
-->>	有箭头的虚线（响应）
-x	末端为叉的实线（表示异步）
--x	末端为叉的虚线（表示异步）

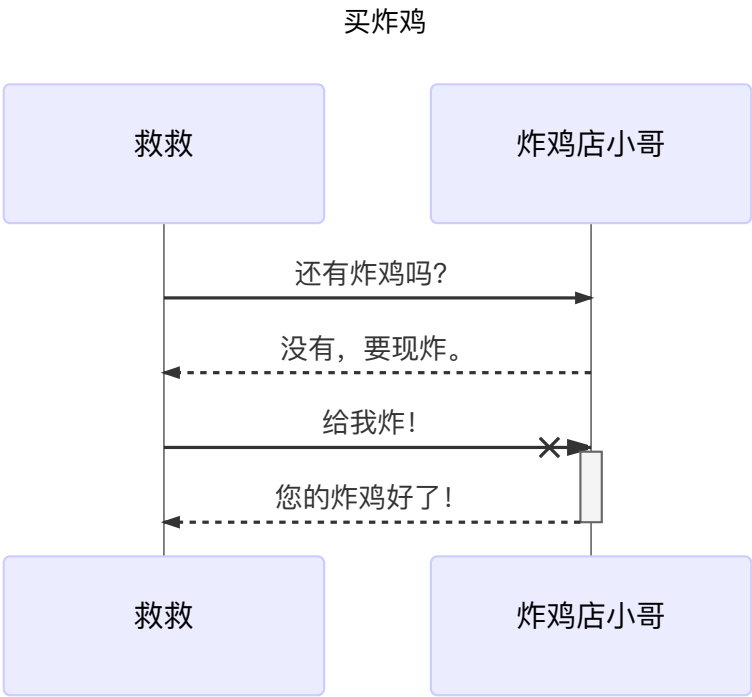
## 处理中-激活框



从消息接收方的时间线上标记一小段时间，表示对消息进行处理的时间间隔。

在消息线末尾增加 `+`，则消息接收者进入当前消息的“处理中”状态； 在消息线末尾增加 `-`，则消息接收者离开当前消息的“处理中”状态。

```
sequenceDiagram
    participant 99 as 救救
    participant seller as 炸鸡店小哥
    99 ->> seller: 还有炸鸡吗?
    seller -->> 99: 没有，要现炸。
    99 -x +seller: 给我炸!
    seller -->> -99: 您的炸鸡好了!
```



注解 (note)

语法如下

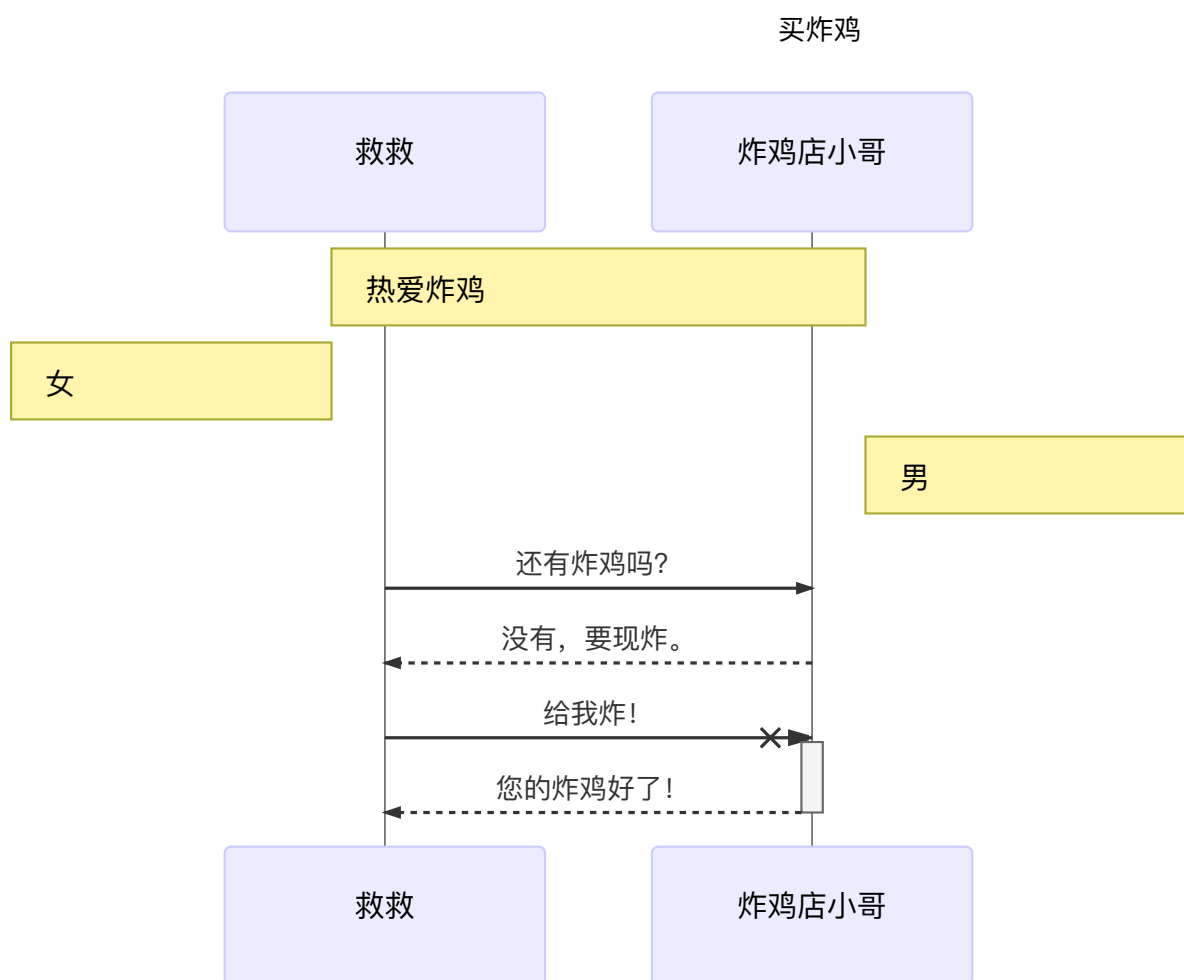
```
Note over 救救, 炸鸡店小哥: 标注文字
```

其中位置表述可以为

表述	含义
right of	右侧
left of	左侧
over	在当中，可以横跨多个参与者

sequenceDiagram

```
participant 99 as 救救
participant seller as 炸鸡店小哥
Note over 99,seller : 热爱炸鸡
Note left of 99 : 女
Note right of seller : 男
99 ->> seller: 还有炸鸡吗?
seller -->> 99: 没有, 要现炸。
99 -x +seller : 给我炸!
seller -->> -99: 您的炸鸡好了!
```

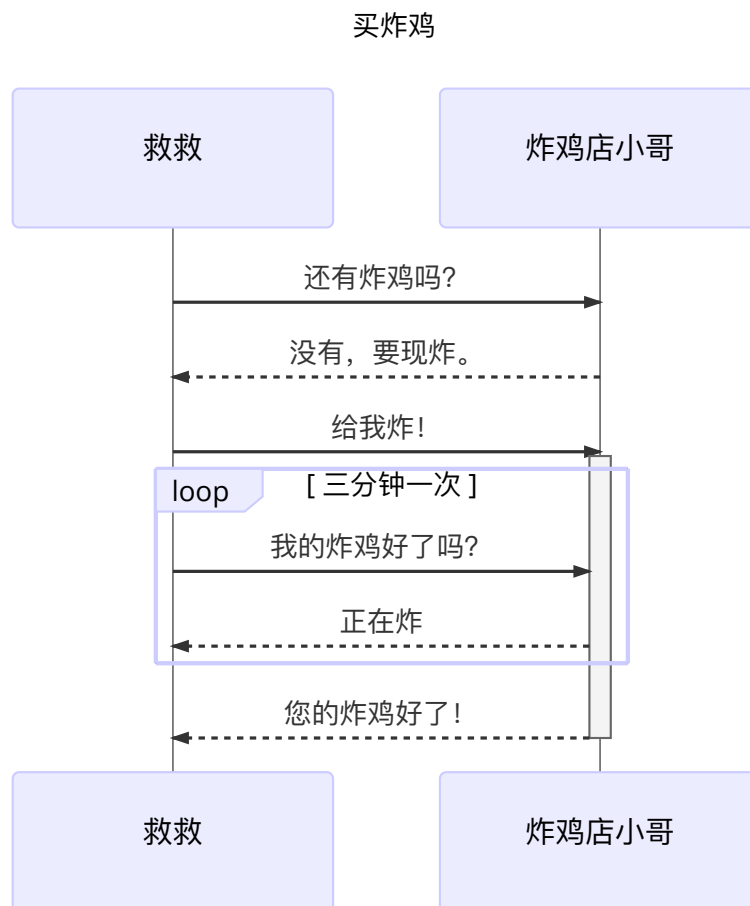


## 循环 (loop)

在条件满足时，重复发出消息序列。（相当于编程语言中的 while 语句。）

```
sequenceDiagram
    participant 99 as 救救
    participant seller as 炸鸡店小哥

    99 ->> seller: 还有炸鸡吗?
    seller -->> 99: 没有, 要现炸。
    99 ->> +seller: 给我炸!
    loop 三分钟一次
        99 ->> seller : 我的炸鸡好了吗?
        seller -->> 99 : 正在炸
    end
    seller -->> -99: 您的炸鸡好了!
```



## 选择 (alt)

在多个条件中作出判断，每个条件将对应不同的消息序列。（相当于 if 及 else if 语句。）

```
sequenceDiagram
    participant 99 as 救救
    participant seller as 炸鸡店小哥
    99 ->> seller : 现在就多少只炸好的炸鸡?
```

```

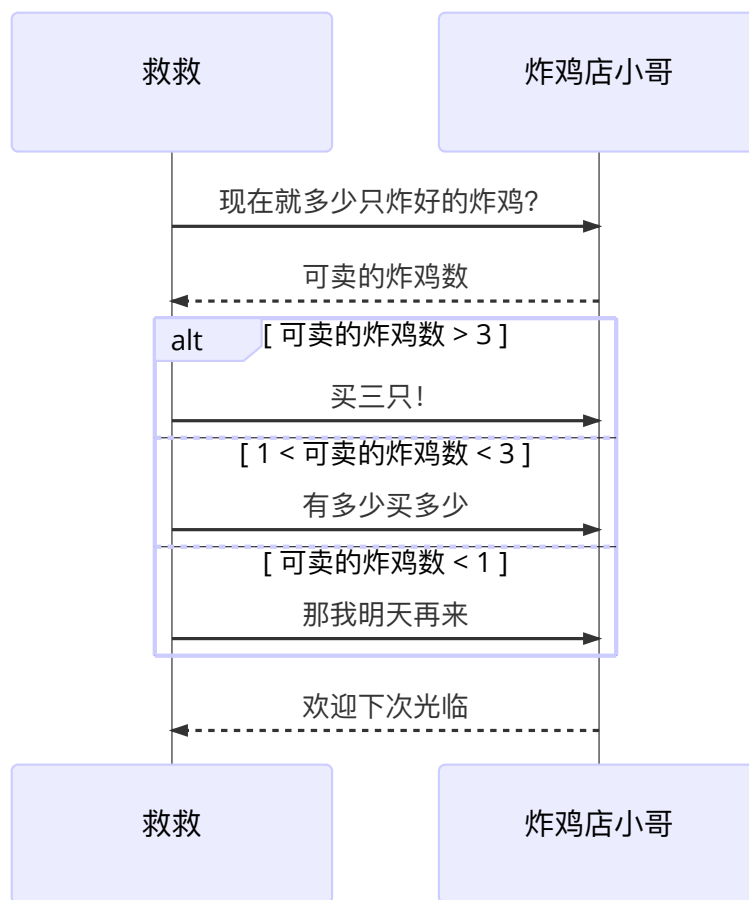
seller -->> 99 : 可卖的炸鸡数

alt 可卖的炸鸡数 > 3
    99 ->> seller : 买三只!
else 1 < 可卖的炸鸡数 < 3
    99 ->> seller : 有多少买多少
else 可卖的炸鸡数 < 1
    99 ->> seller : 那我明天再来
end

seller -->> 99 : 欢迎下次光临

```

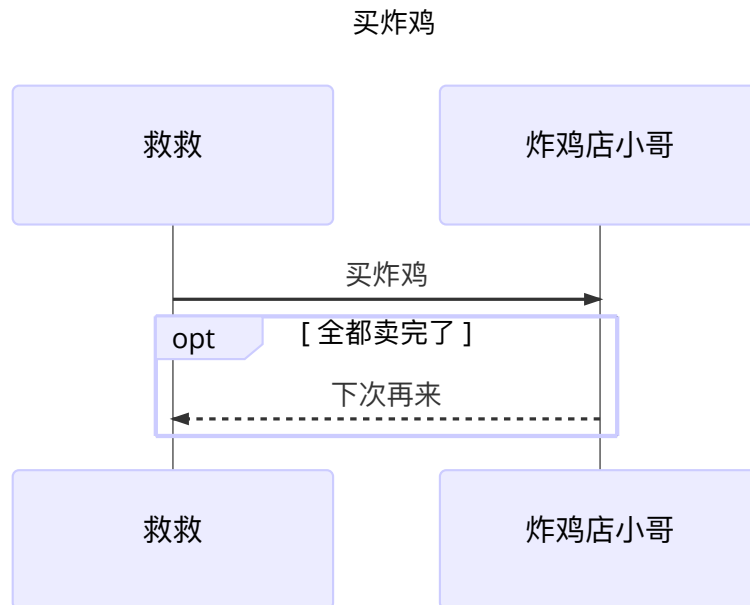
### 买炸鸡



### 可选 (opt)

在某条件满足时执行消息序列，否则不执行。相当于单个分支的 if 语句。

```
sequenceDiagram
    participant 99 as 救救
    participant seller as 炸鸡店小哥
    99 ->> seller : 买炸鸡
    opt 全都卖完了
        seller -->> 99 : 下次再来
    end
```



## 并行 (Par)

将消息序列分成多个片段，这些片段并行执行。

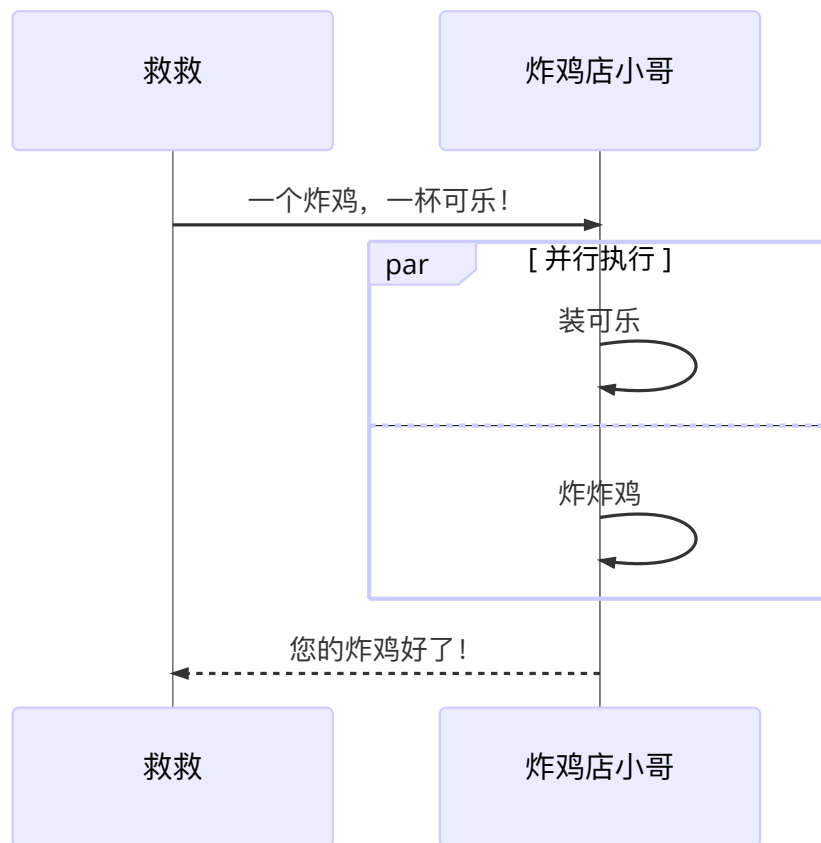
```
sequenceDiagram
    participant 99 as 救救
    participant seller as 炸鸡店小哥

    99 ->> seller : 一个炸鸡，一杯可乐！

    par 并行执行
        seller ->> seller : 装可乐
    and
        seller ->> seller : 炸炸鸡
    end

    seller -->> 99 : 您的炸鸡好了！
```

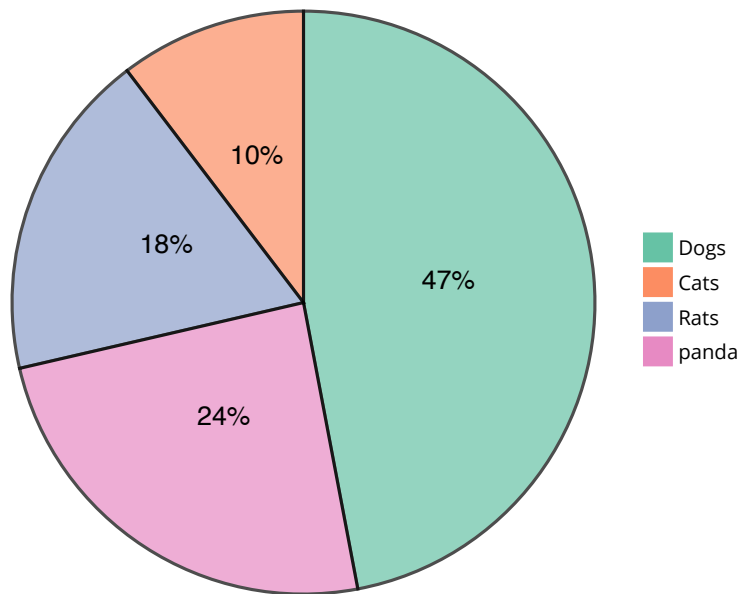
## 买炸鸡



## 饼图 (Pie)

```
pie
    title Pie Chart
    "Dogs" : 386
    "Cats" : 85
    "Rats" : 150
```

Pie Chart



[Typora支持mermaid的官方链接](#)

## 甘特图 (gantt)

```
title 标题
dateFormat 日期格式
section 部分名
任务名:参数一, 参数二, 参数三, 参数四, 参数五

//参数一: crit (是否重要, 红框框) 或者 不填
//参数二: done (已完成)、active (正在进行) 或者 不填(表示为待完成状态)
//参数三: 取小名 或者 不填
//参数四: 任务开始时间
//参数五: 任务结束时间
```

[官方教程](#)

```
gantt
    dateFormat YYYY-MM-DD
    title Adding GANTT diagram functionality to mermaid

    section A section
    Completed task           :done,    des1, 2014-01-06,2014-01-08
    Active task              :active,  des2, 2014-01-09, 3d
    Future task              :         des3, after des2, 5d
    Future task2             :         des4, after des3, 5d

    section Critical tasks
    Completed task in the critical line :crit, done, 2014-01-06,24h
    Implement parser and jison          :crit, done, after des1, 2d
```

```

Create tests for parser           :crit, active, 3d
Future task in critical line     :crit, 5d
Create tests for renderer        :2d
Add to mermaid                   :1d

section Documentation
Describe gantt syntax            :active, a1, after des1, 3d
Add gantt diagram to demo page  :after a1 , 20h
Add another diagram to demo page :doc1, after a1 , 48h

section Last section
Describe gantt syntax            :after doc1, 3d
Add gantt diagram to demo page  :20h
Add another diagram to demo page :48h

```

### Adding GANTT diagram functionality to mermaid

