1. 概述

`Markdown` 语法全部由一些符号组成，其作用往往一目了然。其理念是:让文档更容易读写和修改。

- 通常 `Markdown` 是一种书写格式，而 `HTML` 是一种发布格式。
- `Markdown` 涵盖范围之内的标签都可以在文档中用 `HTML` 撰写，无需额外标注这是HTML还是 `Markdown` ，只需要直接加标签即可, `Markdown` 生成器会自动识别出这些 `HTML` 标签。
    - `HTML` 区块标签（如 `<div>` , `<p>` 等）之间的 `Markdown` 格式语法将不会被处理。
    - `HTML` 行内标签（如 `<span>` , `<cite>` , `<del>` 等）之间的 `Markdown` 格式语法是有效的，会被处理。
    - `HTML` 行内标签（如 `<span>` , `<cite>` , `<del>` 等）可以在 `Markdown` 的段落，列表或者标题内随意使用。

        > 根据个人习惯，你可以不使用 `Markdown` 格式，而完全采用 `HTML` 标签来格式化。

- 在 `HTML` 中， `<` 用于起始标签； `>` 用于结束标签； `&` 用于标记 `HTML` 实体。如果你想显示这几个字符，必须用其实体形式： `&lt;` –> `<` ; `&gt;` –> `>` ; `&amp;` –> `&` 。
- 在 `HTML` 中， `<` , `>` , `&` 的显示形式有 `Markdown` 生成器自动处理。
    - 如果 `&` 是组成HTML实体的字符的一部分，则它按照HTML实体解析。如 `&copy` 解析为 ©。
    - 如果 `&` 不是组成HTML实体的字符的一部分，则它转换成 `&amp;` 。如 `AT&T` 解析为 `AT&amp;T` 。

        > `<` 和 `>` 依此类推。若将其作为 `HTML` 标签的定界符，则 `Markdown` 不做任何转换；否则就自动转换为 `&lt;` , `&gt;` 。

    - 如果在 `Markdown` 的 `code` 元素作用范围内，则 `<` , `>` , `&` 一定会被转成 `&lt;` , `&gt;` 和 `&amp;` 。

2. 区块元素

2.1 段落

一个 `Markdown` 段落是由一个或者多个连续的文本行组成，在其前后要有一个以上的空行(若它的前/后元素为其他区块元素也可以，但两个段落之间必须有空行或者其他区块元素)。

> 空行的定义是：显示上看起来像是空的，比如某一行只包含了空格或者制表符，则即可视为空行。

- 普通段落不能用制表符来缩进
    > 若用制表符缩进，则 `Markdown` 生成器将其解析为 `code` 元素。
- 普通段落不能用空格缩进
    > 若用空格缩进，则在 `HTML` 输出中没有效果。类似于 `HTML` ，在 `Markdown` 中多个空白符只会显示一个空格；若想显示多个空格，参考 `HTML` 。且4个空格会被 `Markdown` 生成器将其解析为 `code` 元素
- `Markdown` 允许段落内插入换行符，然而这并不会在生成的 `HTML` 输出中插入 `<br/>` 标签。要想显示出换行，则有两个方法：
    - 在插入处先按两个以上的空格，然后回车键；而不要直接按回车键
    - 直接在需要换行的地方输入 `<br/>` 标签

2.2 标题

`Markdown` 支持两种标题语法:

> 标题的前后不需要有空行

- 类Setext形式：底线式。用 `=` (最高阶)， `-` (第二阶）。其中 `=` 或者 `-` 的数量至少是一个以上即可。用法为:

    > 为了美观你可以将 `=` 或 `-` 数量延长覆盖标题的所有内容

    ```
    This is H1
    ==========
    ```

    以及

```
This is H2
----------
```

- 类Atx形式：在行首插入1到6个#，对应标题1～6阶。

> 你可以在行尾加上 `#` 来闭合Atx形式的标题，纯粹是为了美观。行尾的 `#` 数量不限。
>
> 通常建议 `#` 和文字之间保留空隔，这是标准用法（某些编辑器只支持标准用法）

```
# This is H1
## This is H2
### This is H3
#### This is H4
##### This is H5
###### This is H6
```

2.3 区块引用

`Markdown` 区块引用使用类似email中的 `>` ，看起来像是你首先断好行，然后在每行的行首添加 `>` 。

```
>This is P1 line1
>, This is P1 line2
>
>This is P2 line1
>, This is P2 line2
```

> 区块引用的前后并不需要有空行。但是有时为为了结束区块引用，需要后置一个空行。

- 你可以只在段落的第一行最前面加上 `>` ，然后并不需要在该段落后面每一行前面加上 `>` 。

```
>This is P1 line1
, This is P1 line2
>
>This is P2 line1
, This is P2 line2
```

- 两段引用之间必须用空行分段，若无空行则视为一段。其中空行可以添加 `>` ，也可以不加 `>` 。
- 区块引用可以嵌套（如引用的引用），只需要根据层次加上不同数量的 `>` 。内层引用是区块元素，自动占用新的段落。

> 必须用空行中断内层区块引用，原因见第一条。

```
>This is Outer line1
>
>> This is Inner line2
>
>This is Outer line3
```

- 区块引用内也可以使用其他的 `Markdown` 元素，包括标题、列表、代码区块等。

```
>## This is H2
>
>1. This is list item1
>2. This is list item2
>
>       This is code(3 Tab or 8  Backspace)
```

2.4 列表

`Markdown` 中的列表分为有序列表和无序列表。

> 列表作为一个区块，其前后和后面必须有空行。

- 无序列表用星号 `*` ,加号 `+` 或者减号 `-` 作为列表标记，标记与内容之间至少一个空格。

```
* Red
* Green
* Blue
```

> 对于 `+` 和 `-` 形式的无序列表，只需要替换掉 `*` 即可。

- 有序列表用数字接着一个英文句点作为标记，标记和内容之间至少一个空格。数字的大小不影响输出的HTML效果。
  > 通常建议数字按照1,2,3...等给出，方便阅读 `Markdown` 源文件。

```
1. Red
2. Green
3. Blue
```

2.4.1 列表项

- 列表中的每一项的内容可以缩进对齐，也可以不对齐。
  - 对齐

    ```
      * item1_line1 is here
        and this is item1_line2
    ```

  - 不对齐

    ```
      * item1_line1 is here
      and this is item1_line2.
    ```

- 列表项之间可以用空行分隔，在输出HTML时， `Markdown` 会将项目内容用 `<p>` 包围起来。

```
* Red

* Green
* Blue
```

的输出 `HTML` 为

```
<ul>
<li><p>Red</p></li>
<li><p>Green</p></li>
<li>Blue</li>
</ul>
```

- 列表项目可以包含多个段落，每个项目下的段落可以最多缩进4个空格或者1个制表符。其中每个段落中的行可以缩进也可以不缩进。

  > 项目中的段落前后不必添加空行
  >
  > 缩进仅仅是为了阅读 `Markdown` 源码方便

```
* this is  item1 p1 line1
and line2.
  this is item1 p2 line1
and line2
* this is item2 p1 line1
  and  line2
  this is item2 p2 line1
  and line2
```

- 列表项目可以包含引用区块，此时 `>` 需要缩进4个空格或者1个制表符。

  > 若没有缩进，则该引用对应的不是该列表项目。

```
* this is item1
    > item1 quote1
    >
    > item1 quote2
```

- 列表项目可以包含代码区块，此时代码区块必须比该列表项多缩进8个空格或者2个制表符。

> 若仅仅缩进4个空格或者1个制表符，则 `Markdown` 生成器认为这是列表项下的段落。

```
* item 1

        code line1
        code line2
```

- `Markdown` 会将以数字加英文点号开头的段落解析为列表，因此对于 `1986. The year is ....` 这种段落，必须通过 `\` 转义: `1986\. The year is ...` 。

2.5 代码区块

在 `Markdown` 中，简单的缩进4个空格或者1个制表符会生成代码区块。它的 `HTML` 输出是用 `<pre>` 和 `<code>` 包围起来的区块。

> 代码区块的前后必须要有空行

- 代码区块会持续到没有缩进的那一行或者下一个区块的开始或者文件结尾。
- 代码区块中的 `&`, `<`, `>` 会自动转成 `HTML` 实体而不是作为 `HTML` 标记。
- 代码区块中的 `Markdown` 语法不会被解析。

代码区块还有一种语法：在代码的两侧均添加三个反引号 ```` ``` ```` 。最好在代码段的首行之前的一行以及代码段的尾行之后的一行添加三个反引号,如：

````
```
code line 1
code line 2
```
````

2.6 分隔线

你可以在一行中用三个以上的星号 `*` ,或者减号 `-` 或者底线 `_` 来建立一个分隔线，行内不能有其他东西。

> 前后不必有空行，除了减号前面必须有空行。这是为了防止出现将作为分隔线的减号 `-` 解析为二阶标题。

- 你可以在星号 `*` 或者减号 `-` 或者底线 `_` 之间插入空格

3. 区段元素

3.1 链接

`Markdown` 支持两种形式的链接语法：行内式和参考式。通常参考式的链接更易于阅读 `Markdown` 源码。

- 行内式的语法为： `[The text of link](http://example.net/ "Title") inline link` ， 输出 `HTML` 为 `<p><a href="http://example.net/" title="Title">The text of link</a> inline link</p>`
    - `](` 之间必须没有空隙，否则 `Markdown` 生成器无法正确解析
    - title属性用双引号或单引号括起来
    - 若url为同主机的资源，则可以用相对路径 `[About](/about)`
- 参考式的语法为： `This is [The text of link][link-id] reference-style link`
    - `[link-id]` 为链接标记，名字由你指定，由字母、数字、空白和标点符号组成，但并不区分大小写
    - `][` 之间最多可以有一个空格，否则 `Markdown` 生成器无法正确解析

  然后在文件的任意地方，你可以将这个标记 `link-id` 的链接内容定义出来。定义语法为: `[link-id]: http//example.com/ "Optinal title here"`
    - 链接内容定义必须在某一行的起始处，或者左侧最多可以有3个空格。若有4个空格则解析为代码区块。
    - `]:` 之间必须没有空格，否则 `Markdown` 生成器无法正确解析
    - title左侧必须有空格，否则就成为了url的一部分
    - title内容可以用双引号或单引号或圆括号等包围
    - url与 `:` 之间可以没有空隔，也可以有空隔
    - tile属性可以放到下一行，也可以增加一些缩进。因为 `Markdown` 中段落中的换行符并不影响 `HTML` 输出

参考式还有一种隐式链接标记：当省略链接标记时，默认链接标记等于链接文字。如 `This is [Google][] website` 等价于 `This is [Google][Google] website` 。

3.2 强调

`Markdown` 用星号 `*` 和底线 `_` 作为标记强调的符号。

- 被星号 `*` 或者底线 `-` 包围的文字，在 `HTML` 中会被 `<em>` 标签包围。如: `*em text*` , `_em text_` 。
- 被两个星号 `*` 或两个底线 `_` 包围的文字，在 `HTML` 中会被 `<strong>` 标签包围。如: `**strong text**` , `__strong text__` 。
- 被三个星号 `*` 或三个底线 `_` 包围的文字，在 `HTML` 中会被 `<strong><em>` 标签包围。
  如: `***strong em text***` , `___strong em text___` 。
- 被四个星号 `*` 或四个底线 `_` 包围的文字，在 `HTML` 中会被 `<strong><strong>` 标签包围。
  如: `****strong strong text****` , `____strong strong text____` 。

其中，星号 `*` 和底线 `_` 与内容之间可以有空隔。

当想在文字某行中同时插入多个星号 `*` 或者同时插入多个底线 `_` 时，需要用反斜线转义。如 `\* insert 2 asterisks \*` 。

3.3 删除线

`Markdown` 中，被两个或两个以上的 `~` 包围的文本或空隔会生成删除线，如 `~~~text~~~` ，其 `HTML` 输出为 ~~text~~ 。

- 删除线的起始 `~` 标记之后的内容必须不能以空隔开始，否则 `Markdown` 生成器无法正确解析
- 删除线的结束 `~` 标记之前的内容最好不以空隔结尾，因为这会让空隔中出现删除线，如 ~~text ~~。

3.4 行内代码

`Markdown` 的行内代码语法简单，即用反引号 ` 来包围行内代码，如 `` `code text` `` 。被包围的文字在 `HTML` 中会被 `<code>` 标签包围。

- 开启代码片段的反引号和结束代码片段的反引号前后可以有空隔，也可以无空隔。
- 代码片段内， `&` , `<` , `>` 等均自动转成HTML实体而不 `HTML` 标记。
- 代码片段内若出现反引号，则要注意:
  - 若想在行内代码中插入反引号，则可以用多个反引号来开启和结束代码片段。如 ``` ``code text`` ```
  - 若开启代码片段用两个反引号，结束代码片段用一个反引号，则相当于向代码中插入一个反引号。如 ``` ``code text ` ```
  - 若开启代码片段用一个反引号，结束代码片段用两个反引号，则 `Markdown` 并不会将它解析为 `<code>` 标签。
  - 若想在代码片段中插入两个反引号，则要留意开启代码片段的反引号之后，结束代码片段的反引号之前要有空隔，
    如 ``` `` `code text` `` ``` 。

3.5 图片

`Markdown` 用一种与链接相似的语法来标记图片。图片同样也允许两种样式:行内式与参考式。

- 行内式语法为： `![Alt text](/path/to/img.jpg "Optional title")` ,其中:
  - `![` 之间必须没有空隔， `](` 之间必须没有空格， `"Optional title"` 与url之间可以有空隔也可以没有空 隔。
  - `"Optional title"` 是可选的，可以没有。
  - title属性用双引号或单引号括起来
- 参考式的语法为： `![Alt text][img-id]` ，其中：
  - `![` 之间必须没有空隔， `][` 之间最多可以有一个空格
  - `img-id` 为图片参考标记，其要求和语法与链接的参考标记一样。语法
    为 `[img-id]: /path/to/img.ipg "Optional title"` 。其中：
    - 链接内容定义必须在某一行的起始处，或者左侧最多可以有3个空格。若有4个空格则解析为代码区块。
    - `]:` 之间必须没有空格，否则 `Markdown` 生成器无法正确解析
    - title内容可以用双引号或单引号或圆括号等包围
    - url与 `:` 之间可以没有空隔，也可以有空隔
    - tile属性可以放到下一行，也可以增加一些缩进。因为 `Markdown` 中段落中的换行符并不影响 `HTML` 输出

目前无法在 `Markdown` 中指定图片的宽高，因此如果有这个需求，则可以用 `HTML` 中的 `<img>` 标签来达到目的。

3.6 自动链接

`Markdown` 中的网址和电子邮箱地址会自动转换成链接。

- 网址的链接文字会与链接地址相同，如 `http://www.google.com` 的 `HTML` 输出为 `<a href="http://www.google.com">http://www.google.com</a>`
- 邮件的自动链接会有一个编码转换过程，这是为了垃圾邮件机器人抓取邮箱地址。如 `huaxz1986@163.com` 的 `HTML` 输出为

```
  <a href="&#109;&#97;&#105;&#108;&#x74;&#x6f;&#58;&#104;&#117;&#x61;&#120;&#122;&#x31;&#57;&#x38;&#54;&#64;&#49;&#x36;&#x33;&#x2e;&#99;&#x6f;&#x6d;">&#x68;&#x75;&#97;&#120;&#122;&#x31;&#x39;&#x38;&#x36;&#x40;&#49;&#x36;&#51;&#46;&#x63;&#111;&#109;</a>
```

3.7 脚注

`Markdown` 中脚注的语法是: `text [^note_id]` 。

- `text` 与 `[` 之间可以有空隔，也可以无空隔。若有空隔，则 `HTML` 输出文档中脚注的链接文字与 `text` 之间也会有空格。
- `[^` 之间必须没有空隔，否则 `Markdown` 生成器无法正确解析
- `^` 与 `]` 之间出现的文字都是 `note_id` 的组成部分，包括空隔字符，因此 `note_id` 最好不要以空隔结尾

在当前 `Markdown` 文件的某个地方，给出 `note_id` 的定义，语法是: `[^note_id]: foot_note descrption text` 。

- 必须有 `note_id` 的定义，脚注的语法才完整，否则 `Markdown` 生成器无法正确解析
- `[^` 之间必须没有空隔，否则 `Markdown` 生成器无法正确解析
- `^` 与 `]` 之间出现的文字都是 `note_id` 的组成部分，包括空隔字符，因此 `note_id` 最好不要以空隔结尾
- `]:` 之间必须没有空格，否则 `Markdown` 生成器无法正确解析
- `:` 与 `foot_note descrption text` 之间可以有空隔，也可以没有空隔

3.8 转义字符

`Markdown` 支持利用反斜线 `\` 来转义字符。如 `\#` 会转义作为标题起始标记的 `#` 字符。

3.9 首行缩进

`Markdown` 可以利用 `HTML` 中的 ` ` 在输出的 `HTML` 中产生一个空格符，利用 ` ` 在输出的 `HTML` 中产生两个空格符。
如 ` first line` 的 `HTML` 输出为 first line， ` first line` 的 `HTML` 输出为 first line。

3.10 技巧

- 添加空行可以结束当前的格式状态从而启用新的格式状态，建议改变格式时均添加一个空行
- 普通段落尽量不用空隔或制表符来缩进，即使其缩进后的 `HTML` 输出结果看起来是对的

4 `Markdown` 的表格扩展

目前大多数 `Markdown` 编辑器支持 `Markdown` 的表格扩展，其语法为：

```
    |   head1   |   head2   |   head3   |  head4    |
    |    ---    |   :---    |   :---:   |   ---:    |
    | row1col1 | row1col2 | row1col3 | row1col4 |
    | r2c1      | r22       | r23       | r24       |
```

其显示结果为:

head1	head2	head3	head4
row1col1	row1col2	row1col3	row1col4
r2c1	r22	r23	r24

- 第一行为标题；第二行为格式说明，第三行开始为数据单元格
- 第二行格式说明给出每一列的格式。
  - `---` 表示没有任何格式设置
  - `:---` 表示左对齐

    - `:---:` 表示居中对齐
    - `---:` 表示右对齐

- 第二行格式说明中可以只使用一个 `-` 或者更多个 `-`
- 单元格内部可以添加空格也可以不添加空格
- 无法调整表格的宽高，也无法设置单元格的样式

5 cutemarked数学公式扩展

cutemarked开源 `Markdown` 编辑器支持数学公式扩展，它利用了 `MathJax` 这个 `JavaScript` 引擎来渲染数学公式。cutemarked支持代码高亮，支持数学公式，通过在"设置"菜单中可以选择打开或者关闭这两个功能。

`MathJax` 支持以 `LATEX` 格式作为输入。在cutemarked的源文件中，与数学输入相关的源代码有：

- app/template_presentation.html中的

```
  <script>
  Reveal.initialize({
      math: {
      mathjax: 'https://cdn.mathjax.org/mathjax/latest/MathJax.js',
      config: 'TeX-AMS_HTML-full'  // See http://docs.mathjax.org/en/latest/config-   files.html
      },
      dependencies: [
      { src: 'https://cdn.jsdelivr.net/reveal.js/2.6.2/plugin/markdown/marked.js', condition: function(
) { return !!document.querySelector( '[data-markdown]' ); } },
      { src: 'https://cdn.jsdelivr.net/reveal.js/2.6.2/plugin/markdown/markdown.js', condition: functio
n() { return !!document.querySelector( '[data-markdown]' ); } },
      <!--__REVEAL_PLUGINS__-->
  ]
  });
  </script>
```

- app-static/template/presentationtemplate.cpp中的

```
  QString PresentationTemplate::buildRevealPlugins(RenderOptions options) const
  {
      QString plugins;

      // add MathJax.js script as reveal plugin
      if (options.testFlag(Template::MathSupport)) {
          plugins += "{ src: 'https://cdn.jsdelivr.net/reveal.js/2.6.2/plugin/math/math.js', async: tru
e },\n";
      }
    ...
```

- app-static/template/htmltemplate.cpp中的

```
  QString HtmlTemplate::buildHtmlHeader(RenderOptions options) const
  {
      QString header;
      ...
      // add MathJax.js script to HTML header
      if (options.testFlag(Template::MathSupport)) {
          // Add MathJax support for inline LaTeX Math
          if (options.testFlag(Template::MathInlineSupport)) {
              header += "<script type=\"text/x-mathjax-config\">MathJax.Hub.Config({tex2jax: {inlineMat
h: [['$','$'], ['\\\\(','\\\\)']]}});</script>";
          }
          header += "<script type=\"text/javascript\" src=\"http://cdn.mathjax.org/mathjax/latest/MathJ
ax.js?config=TeX-AMS-MML_HTMLorMML\"></script>\n";
      }
      ...
```

cutemarked与 `LATEX` 的语法存在以下区别：

- cutemarked中，对于简单的上标，只需要输入 `x^2` ，而不需要任何界定符;而 `LATEX` 中需要使用 `$ x^2 $` 。
- cutemarked中，若一行 `$$...$$` 公式中命令太多可能出现解析失败的情况， 此时需要拆分 `$$...$$` 到多个 `$$...$$` 中；若仍然无法解析则需要在 `$$...$$` 之间添加空行

- cutemarked由于对某些反斜线 `\` 进行了转义，因此出现在数学公式中的反斜线需要注意。
  - 若公式中的 `\` 后面紧跟非特殊字符，则cutemarked并不执行转义。 如cutemarked中输入 `\begin` ，则 `MathJax` 引擎接收到的字符是 `\begin` 。
  - 若公式中的 `\` 后面紧跟特殊字符，则cutemarked执行转义。 如cutemarked中输入 `\\` ，则 `MathJax` 引擎接收到的字符是 `\` 。

! 以下的所有公式均是针对cutemarked的语法

5.1 公式类型

数学公式有两类，一种是以区块的形式显示，称为区块公式；一种是在行内显示，称为行内公式。

- 区块公式自成一段显示并居中对齐，哪怕它位于文本行中,它的定界符为:
  - 以 `$$...$$` 作为定界符
  - 以 `\begin{equation}...\end{equation}` 作为定界符

    > 在标准的 `LATEX` 中，该定界符会在公式右侧自动添加公式的编号如 `1.1` , `1.2` 等。 如果不想自动编号则用 `\begin{equation*}...\end{equation*}`
    >
    > 还有一些 `\begin{...}...\end{...}` 类型的定界符。这些定界符也可以出现在 `$$...$$` 内部， 此时作用类似于分组

  - 以 `\\[...\\]` 作为定界符

    > `LATEX` 中是以 `\[...\]` 作为定界符； 该定界符仅仅是 `\begin{equation*}...\end{equation*}` 的语法糖

- 行内公式以 `\\( ...\\)` 作为定界符，它显示较小并且在行中显示，如$\sum_1^n$。

  > 在 `LATEX` 中是以 `\(...\)` 作为行内公式定界符

对于部分公式，其行内的显示与区块中的显示会有所不同。行内式：$\sum_1^n \lim_{1\to n} \prod_1^n$。区块式:

$$\sum_1^n \lim_{1\to n} \prod_1^n$$

5.2 公式大小

`LATEX` 公式有四种大小，依次为displaystyle, textstyle, scriptstyle, scriptscriptstyle。例：

$$\frac{1}{8} \ \frac{1}{8} \ \frac{1}{8} \ \frac{1}{8}$$

通常区块样式默认使用displaystyle，行内样式使用textstyle，上下标使用scriptstyle。你也可以在公式中使用下列命令显式调整公式整体大小或者部分公式大小：

- `\displaystyle{...}` ：将 `{...}` 中的公式显式设置为displaystyle
- `\textstyle{...}` ：将 `{...}` 中的公式显式设置为textstyle
- `\scriptstyle{...}` ：将 `{...}` 中的公式显式设置为scriptstyle
- `\scriptscriptstyle{...}` ：将 `{...}` 中的公式显式设置为scriptscriptstyle

5.3 间隔与换行

`LATEX` 通过内部策略管理公式内部的空间，手工输入的空隔和换行并不能影响公式的 `HTML` 输出。 如 `\\(ab\\)` 与 `\\(a b\\)` 均显示为 ab。

5.3.1 间隔

为了增加公式内部的空隔，可以使用以下命令

- `\,` 增加$\frac{3}{18}$个字符M的宽度
- `\\:` 增加$\frac{4}{18}$个字符M的宽度

  > 在 `LATEX` 中是以 `\:` 作为$\frac{4}{18}$个字符M的宽度

- `\` （反斜线后跟一个空格符）增加1个字符M的宽度
- `\quad` 增加更大的宽度
- `\qquad` 增加最大的宽度
- `\\!` 会产生负的$\frac{3}{18}$个字符M的宽度

> 在 `LATEX` 中是以 `\!` 作为 $-\frac{3}{18}$ 个字符M的宽度

$$
\begin{aligned}
ab &: \text{no space} \\
a\,b &: \backslash, \\
a\:b &: \backslash: \\
a\ b &: \backslash \\
a\quad b &: \backslash\text{quad} \\
a\qquad b &: \backslash\text{qquad} \\
a\!b &: \backslash!
\end{aligned}
$$

5.3.2 换行

对于多行公式，简单的回车符并不能达到换行的效果，必须通过命令 `\\\` 实现公式换行。

> 在 `LATEX` 中是以 `\\` 作为换行命令

如 `$$ abcd\\\efg$$` 的 `HTML` 输出为：

$$
\begin{aligned}
abcd \\
efg
\end{aligned}
$$

这里有个问题，无法调整行间距。可以通过命令 `\\\\[2ex]` 实现公式换行且将间距设置为2ex， 其中1ex相当于原始的间距，2ex为2倍原始间距....。

> 在 `LATEX` 中是以 `\\[2ex]` 作为换行加设置间距的命令

如 `$$ abcd\\\\[2ex]efg$$` 的 `HTML` 输出为：

$$
\begin{aligned}
abcd \\[2ex]
efg
\end{aligned}
$$

5.3.3 对齐

- 在 `LATEX` 中区块公式中的多行对齐使用 `\begin{align}...\end{align}` 命令，其中在公式中利用 `&` 锚定需要对齐的位置。
  如 `\begin{align} a &=b+c \\\ &=d+e\end{align}` 的 `HTML` 输出为：

$$
\begin{aligned}
a &= b+c \\
&= d+e
\end{aligned}
$$

- 有的时候公式太长，则需要对公式折叠成多行。此时有几个原则:
  - 每行必须以操作数结束，而不能以等号或者操作符结束
  - 尽量不要破坏高优先级的计算单元，如不要出现:

$$
\begin{aligned}
a &= b+c-d \\
&\div e+f
\end{aligned}
$$

- 多行对齐还可以使用 `\begin{eqnarray}...\end{eqnarray}` 命令来同时对齐和折叠，
  如 `\begin{eqnarray}a&=&b+c \\\ &=&d+e+f+g+h+i+j+k+k \\\ && +m+n+o \end{eqnarray}` 的 `HTML` 输出为：

$$
\begin{aligned}
a &= b+c \\
&= d+e+f+g+h+i+j+k+k \\
&\quad +m+n+o
\end{aligned}
$$

> 它与 `\begin{align}...\end{align}` 的区别在于前者可以让换行后的 $+m+n+o$ 与 d 对齐，而不是与 $=$ 对齐

5.3.4 上下标对齐

`LATEX` 中使用 `\phantom` 实现上下标的垂直对齐。例如：

- `\\({}^{14}_6 \text{C}\\)` 的 `HTML` 输出为：$_{6}^{14}\text{C}$
- `\\({}^{14}_{\phantom{1}6} \text{C}\\)` 的 `HTML` 输出为：$_{6}^{14}\text{C}$

5.3.5 上下标折叠

当上标或者下标比较长的时候，可以用 `\substack` 命令折叠。如
`$$\sum^{\substack{n\neq k \\\ n<100}}_{\substack{0<i<n \\\ j\subseteq i}}$$` 的 `HTML` 输出为：

$$
\sum^{\substack{n\neq k \\ n<100}}_{\substack{0<i<n \\ j\subseteq i}}
$$

> 若无 `\substack` 命令，则上下标不会换行，它们只能各自占据一行显示

5.4 上标与下标

`LATEX` 中的上标用 `^` 命令，下标用 `_` 命令。默认情况下上/下标仅对下一个组起作用。由于cutemarked中，对于简单的上标，只需要输入 `x^2` ，而不需要任何界定符;而 `LATEX` 中需要使用 `$ x^2 $` 。因此对于区块公式和行内公式中的上标，必须用 `^{}` 命令，即必须用 `{}` 分组，否则 `MathJax` 解析失败。

> `LATEX` 的组是以 `{}` 包裹起来的内容，若无 `{}` 则为单个字符

当上下标的内容多于一个字符时，必须用 `{}` 将其包裹起来作为一个分组。上下标是可以相互嵌套使用的。例子：

$$x \quad x_i \quad x^2 \quad x_i^2$$

$$x_{i_i} \quad x^{2^2}$$

$$x^{2_i} \quad x_{i^2}$$

- 你也可以在左边添加上下标。如 `\\( {}^{2}_i C\\)` 的 `HTML` 输出为 ${}^{2}_{i}C$
- 你也可以利用 `\sideset` 命令在两边添加上下标。如 `\\( \sideset{^2_i}{^n_m} C\\)` 的 `HTML` 输出为 $\sideset{^2_i}{^n_m}C$

> 你无法使用 `\\( {}^{2}_i C ^{n}_m\\)` 来构造两边的上下标

5.5 括号

- `LATEX` 中的小括号 `()` 与中括号 `[]` 使用原始的小括号 `()` 与中括号 `[]` 即可。
- 由于大括号 `{}` 是分组命令，需要用 `\` 转义。因此在cutemarked中，用 `\\{ \\}` 代表大括号， 如 `\\( \\{x+y \\} \\)` 的 `HTML` 输出为 $\{x+y\}$ 。

> `LATEX` 中的大括号用 `\{ \}` 表示，因为cutemarked将 `\\{` 转义成 `\{` 传递给 `MathJax` 引擎。
>
> `LATEX` 中还可以用 `\lbrace` 和 `\rbrace` 命令表示大括号， 如 `\\( \lbrace x+y \rbrace \\)` 的 `HTML` 输出为 $\lbrace x+y \rbrace$ 。

- 尖括号使用 `\langle` 和 `\rangle` 表示。如 `\\(\langle x+y \rangle \\)` 的 `HTML` 输出为 $\langle x+y \rangle$

> `LATEX` 中不能直接用 `<>` 表示尖括号，下面的例子可以看出区别：
>
> `\\( <x+y>\\)` 的 `HTML` 输出 $<x+y>$
>
> `\\(\langle x+y \rangle \\)` 的 `HTML` 输出为 $\langle x+y \rangle$

- 上取整符号用 `\lceil` 和 `\rceil` 表示。如 `\\(\lceil x+y \rceil \\)` 的 `HTML` 输出为 $\lceil x+y \rceil$
- 下取整符号用 `\lfloor` 和 `\rfloor` 表示。如 `\\(\lfloor x+y \rfloor \\)` 的 `HTML` 输出为 $\lfloor x+y \rfloor$
- 单竖线使用 `|...|` 即可。如 `\\( | x+y | \\)` 的 `HTML` 输出为 $|x+y|$
- 双竖线使用 `\\|...\\|` 即可。如 `\\( \\| x+y \\| \\)` 的 `HTML` 输出为 $\|x+y\|$

> `LATEX` 中双竖线使用 `\|...\|` 命令

- 向下箭头使用 `\Downarrow...\Downarrow` 即可。如 `\\( \Downarrow x+y \Downarrow \\)` 的 `HTML` 输出为 $\Downarrow x+y \Downarrow$

> 还有很多类似的符号

5.5.1 自适应大小的括号

上述括号有一个缺点：无法根据公式的高度进行缩放。如 `$$ (\frac{1}{16}) $$` 的 `HTML` 输出为

$$(\frac{1}{16})$$

可以使用 `\left...\right` 命令添加自适应大小的括号。

- 自适应大小的小括号使用 `\left(...\right)` 命令，如 `$$\left(\frac{1}{16}\right)$$` 的 `HTML` 输出为

$$\left(\frac{1}{16}\right)$$

- 自适应大小的中括号用 `\left[...\right]` 命令， 如 `$$\left[\frac{1}{16}\right]$$` 的 `HTML` 输出为

$$\left[\frac{1}{16}\right]$$

- 自适应大小的大括号用 `\left\\{...\right\\}` 命令， 如 `$$\left\\{\frac{1}{16}\right\\}$$` 的 `HTML` 输出为

$$\left\{\frac{1}{16}\right\}$$

> `LATEX` 中的自适应大小的大括号命令为 `\left\{...\right\}`

- 自适应大小的尖括号使用 `\left<...\right>` 命令，如 `$$\left<\frac{1}{16}\right>$$` 的 `HTML` 输出为

$$\left<\frac{1}{16}\right>$$

> 也可以使用 `\left\langle...\right\rangle` 命令，它与 `\left<...\right>` 显示相同

- 自适应大小的上取整符号使用 `\left\lceil...\right\rceil` 命令，
  如 `$$\left\lceil\frac{1}{16}\right\rceil$$` 的 `HTML` 输出为

$$\left\lceil\frac{1}{16}\right\rceil$$

- 自适应大小的下取整符号使用 `\left\lfloor...\right\rfloor` 命令，
  如 `$$\left\lfloor\frac{1}{16}\right\rfloor$$` 的 `HTML` 输出为

$$\left\lfloor\frac{1}{16}\right\rfloor$$

- 自适应大小的单竖线使用 `\left|...\right|` 命令， 如 `$$\left|\frac{1}{16}\right|$$` 的 `HTML` 输出为

$$\left|\frac{1}{16}\right|$$

- 自适应大小的双竖线使用 `\left\\|...\right\\|` 命令， 如 `$$\left\\|\frac{1}{16}\right\\|$$` 的 `HTML` 输出为

$$\left\|\frac{1}{16}\right\|$$

> `LATEX` 中自适应大小的双竖线使用 `\left\|...\right\|` 命令

- 自适应大小的向下箭头使用 `\left\Downarrow...\right\Downarrow` 命令，
  如 `$$\left\Downarrow\frac{1}{16}\right\Downarrow$$` 的 `HTML` 输出为

$$\left\Downarrow\frac{1}{16}\right\Downarrow$$

> `还有很多类似的符号

上述所有的命令都可以仅仅使用半边符号，例如 `\\( \left( x+y \right. \\)` 的 `HTML` 输出为$\left(x+y \right.$。其语法在于：对丢弃的那半边符号设为 `.`（英文句点） 注意：

- 所有的自适应大小的括号， `\left` 和 `\right` 必须成对出现
- 所有的 `\left` 和 `\right` 组合必须匹配
  - 要么二者都是相同类型的括号
  - 要么其中有一个是英文句点 `.`，此时为单边自适应大小的括号

5.5.2 手动设置不同大小的括号

可以使用 `\big`， `\Big`, `\bigg`, `\Bigg` 命令手动设置不同大小的括号，其尺寸以此递增。如

```
$$
\Bigg(\bigg(\Big(\big(  (\frac{1}{16}) \big)\Big)\bigg)\Bigg) \\
\Bigg[\bigg[\Big[\big[  [\frac{1}{16}] \big]\Big]\bigg]\Bigg] \\
\Bigg\{\bigg\{\Big\{\big\{  \{\frac{1}{16}\} \big\}\Big\}\bigg\}\Bigg\}\\
\Bigg|\bigg|\Big|\big|  |\frac{1}{16}| \big|\Big|\bigg|\Bigg|
$$
```

的 `HTML` 输出为：

$$
\Bigg(\bigg(\Big(\big((\frac{1}{16}) \big)\Big)\bigg)\Bigg) \\
\Bigg[\bigg[\Big[\big[[\frac{1}{16}] \big]\Big]\bigg]\Bigg] \\
\Bigg\{\bigg\{\Big\{\big\{ \{\frac{1}{16}\} \big\}\Big\}\bigg\}\Bigg\}\\
\Bigg|\bigg|\Big|\big| |\frac{1}{16}| \big|\Big|\bigg|\Bigg|
$$

5.6 `math` 模式和文本模式

`LATEX` 中的公式有两种模式： `math` 模式和文本模式。在 `math` 模式中，所有的换行与空隔均不起作用，公式的排版布局由公式的表达式自己定义，也可以由某些命令如 `\,` 等决定。参见前面的“换行与空格”章节。

在 `math` 模式中，每个字母都会被认为是变量名，会按照变量名的样式进行显示。如果你想在公式中输入文本，使用命

令 `\text{this is text}` 。如 `\\(this is math mode \quad \text{this is text mode} \\)` 的 `HTML` 输出为 $thisismathmode \quad \text{this is text mode}$。

- 如果想在 `\text` 模式中以 `math` 模式显示变量，则可以用 `$...$` 命令包裹这个变量名。
  如 `\\(\text{this is text mode and $var$} \\)` 的 `HTML` 输出为 $\text{this is text mode and } var$。

5.7 字体

`LATEX` 中公式的字体可以通过下列命令设置。

- 用 `\mathbb {...}` 或者 `\Bbb {...}` 来设置黑版粗体，这种字体常用于表示实数R，整数N，有理数Q，复数Z。
  如 `\\(normalx \quad \mathbb{R,N,Q,Z}\\)` 的 `HTML` 输出为：$normalx \quad \mathbb{R,N,Q,Z}$
- 用 `\mathbf` 来设置黑体。如 `\\(normalx \quad \mathbf{X,Y,M,N}\\)` 的 `HTML` 输出为 $normalx \quad \mathbf{X,Y,M,N}$
- 用 `\mathtt` 来设置打印字体。如 `\\(normalx \quad \mathtt{X,Y,M,N}\\)` 的 `HTML` 输出为 $normalx \quad \mathtt{X,Y,M,N}$
- 用 `\mathrm` 来设置罗马字体（即英文中的Times New Roman字体）。 如 `\\(normalx \quad \mathrm{X,Y,M,N}\\)` 的 `HTML` 输出为$normalx \quad \mathrm{X,Y,M,N}$
- 用 `\mathit` 来设置italic字体。 如 `\\(normalx \quad \mathit{X,Y,M,N}\\)` 的 `HTML` 输出为 $normalx \quad \mathit{X,Y,M,N}$
- 用 `\mathcal` 来设置花体。 如 `\\(normalx \quad \mathcal{X,Y,M,N}\\)` 的 `HTML` 输出为 $normalx \quad \mathcal{X,Y,M,N}$
- 用 `\mathsf` 来设置等线体。 如 `\\(normalx \quad \mathsf{X,Y,M,N}\\)` 的 `HTML` 输出为 $normalx \quad \mathsf{X,Y,M,N}$
- 用 `\mit` 来设置数学斜体。 如 `\\(normalx \quad \mit{X,Y,M,N}\\)` 的 `HTML` 输出为 $normalx \quad \mit{X,Y,M,N}$
- 用 `\mathscr` 来设置手写体。如 `\\(normalx \quad \mathscr{X,Y,M,N}\\)` 的 `HTML` 输出为 $normalx \quad \mathscr{X,Y,M,N}$
- 用 `\mathfrak` 来设置Fraktur字体（一种德国体）。如 `\\(normalx \quad \mathfrak{X,Y,M,N}\\)` 的 `HTML` 输出为 $normalx \quad \mathfrak{X,Y,M,N}$

对于以 `\math...` 开始的命令均有简写形式，其简写形式为去掉 `math` 的简写，除了 `\mathbb` 的简写为 `\Bbb` 来设置黑版粗体。如：

```
\begin{align}
\Bbb{R,N,Q,Z,A,B,C}  \quad &:\text{\Bbb}\\\
\bf{R,N,Q,Z,A,B,C}  \quad &:\text{\bf}\\\
\tt{R,N,Q,Z,A,B,C}  \quad&:\text{\tt} \\\
\rm{R,N,Q,Z,A,B,C}  \quad &:\text{\rm}\\\
\it{R,N,Q,Z,A,B,C}  \quad &:\text{\it}\\\
\cal{R,N,Q,Z,A,B,C}  \quad &:\text{\cal}\\\
\sf{R,N,Q,Z,A,B,C}  \quad &:\text{\sf}\\\
\scr{R,N,Q,Z,A,B,C}   \quad&:\text{\scr}\\\
\frak{R,N,Q,Z,A,B,C}  \quad &:\text{\frak}\\\
\end{align}
```

的 `HTML` 输出为：

$$
\begin{aligned}
\Bbb{R,N,Q,Z,A,B,C} \quad &:\text{\Bbb}\\
\bf{R,N,Q,Z,A,B,C} \quad &:\text{\bf}\\
\tt{R,N,Q,Z,A,B,C} \quad&:\text{\tt} \\
\rm{R,N,Q,Z,A,B,C} \quad &:\text{\rm}\\
\it{R,N,Q,Z,A,B,C} \quad &:\text{\it}\\
\cal{R,N,Q,Z,A,B,C} \quad &:\text{\cal}\\
\sf{R,N,Q,Z,A,B,C} \quad &:\text{\sf}\\
\scr{R,N,Q,Z,A,B,C}   \quad&:\text{\scr}\\
\frak{R,N,Q,Z,A,B,C} \quad &:\text{\frak}\\
\end{aligned}
$$

5.8 公式内转义

在 `LATEX` 公式内部，可以通过反斜线 `\` 来转义字符。如 `` `\\( a\\_1 \\) `` 的 `HTML` 输出为 a_1

> cutemarked中输入 `\\` ，则 `MathJax` 引擎接收到的字符是 `\` ，因为cutemarked已经执行了一层转义

5.9 表格

在 `LATEX` 公式中，通过 `\begin{array}{列样式}...\end{array}` 命令来创建表格。其中表格各行用 `\\` 分隔，各列用 `&` 分隔；列样式需要依次给出每一列的样式： `c` 表示居中对齐， `l` 表示左对齐， `r` 表示右对齐，若有竖线表示引入一条列竖线。如

```
\begin{array} {c|lrc}
n & \text{Left} & \text{Right} &\text{Center} \\\
1 & \hline 0.24 &  1   \\\
2 & -1 &  & -8\\\
3 & -20 & 200 & 1+10i
\end{array}
```

的 `HTML` 输出为：

n	Left	Right	Center
1	0.24	1	
2	−1		−8
3	−20	200	$1+10i$

> cutemarked中需要输入 `\\\` ，则 `MathJax` 引擎接收到的字符是 `\\` ，因为cutemarked已经执行了一层转义

- 可以在行中添加 `\hline` 从而在本行前面加入一条直线， `\hline` 添加的位置必须是该行某个数据之前， 而不能是 `&` 之前
- 若某一行的数据较少，则依次从左到右填充（如前例中的第一行）
- 若两个列分隔符 `&` 之间为空，则该单元格为空（如前例中的第二行）
- 可以定义表格的嵌套，即表格的某个单元格也是表格

5.10 矩阵

在 `LATEX` 中，通过 `\begin{matrix}...\end{matrix}` 命令来创建矩阵。其中矩阵各行用 `\\` 分隔，各列用 `&` 分隔。如：

```
\begin{matrix}
1 & x & x^{2} \\
1 & y & y^{2} \\
1 & z & z^{2}
\end{matrix}
```

的 `HTML` 输出为：

$$\begin{matrix}
1 & x & x^{2} \\
1 & y & y^{2} \\
1 & z & z^{2}
\end{matrix}$$

> cutemarked中需要输入 `\\\` ，则 `MathJax` 引擎接收到的字符是 `\\` ，因为cutemarked已经执行了一层转义

- 可以用 `\cdots` ⋯ , `\ddots` ⋱ , `\vdots` ⋮ 作为矩阵的元素，如:

  ```
  \begin{matrix}
  1 & a_1 & a_1^{2} & \cdots & a_1^{n} \\
  1 & a_2 & a_2^{2} & \cdots & a_2^{n} \\
  \vdots &\vdots &\vdots &\ddots &\vdots \\
  1 & a_m & a_m^{2} & \cdots & a_m^{n}
  \end{matrix}
  ```

  的 `HTML` 输出为：

$$\begin{matrix}
1 & a_1 & a_1^{2} & \cdots & a_1^{n} \\
1 & a_2 & a_2^{2} & \cdots & a_2^{n} \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
1 & a_m & a_m^{2} & \cdots & a_m^{n}
\end{matrix}$$

5.10.1 带括号的矩阵

可以对矩阵加上括号，采用 **5.5.1** 节的方法，利用 `\left...\right` 包裹 `\begin{matrix}...\end{matrix}` 命令的方法添加自适应括号。或者采用特殊的命令，这些命令将生成特殊的带括号的矩阵。

- `\begin{pmatrix}...\end{pmatrix}` 生成带小括号的矩阵，如：

  ```
  \begin{pmatrix}
  1 & 2 \\
  3 &4
  \end{pmatrix}
  ```

  的 `HTML` 输出为：

$$\begin{pmatrix}
1 & 2 \\
3 &4
\end{pmatrix}$$

- `\begin{Bmatrix}...\end{Bmatrix}` 生成带大括号的矩阵，如：

  ```
  \begin{Bmatrix}
  1 & 2 \\
  3 &4
  \end{Bmatrix}
  ```

的 `HTML` 输出为：

$$\begin{Bmatrix} 1 & 2 \\ 3 & 4 \end{Bmatrix}$$

- `\begin{bmatrix}...\end{bmatrix}` 生成带中括号的矩阵，如：

```
\begin{bmatrix}
1 & 2 \\
3 &4
\end{bmatrix}
```

的 `HTML` 输出为：

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$$

- `\begin{vmatrix}...\end{vmatrix}` 生成带单竖线的矩阵，如：

```
\begin{vmatrix}
1 & 2 \\
3 &4
\end{vmatrix}
```

的 `HTML` 输出为：

$$\begin{vmatrix} 1 & 2 \\ 3 & 4 \end{vmatrix}$$

- `\begin{Vmatrix}...\end{Vmatrix}` 生成带双竖线的矩阵，如：

```
\begin{Vmatrix}
1 & 2 \\
3 &4
\end{Vmatrix}
```

的 `HTML` 输出为：

$$\begin{Vmatrix} 1 & 2 \\ 3 & 4 \end{Vmatrix}$$

5.10.2 增广矩阵

对于增广矩阵，不能使用 `\begin{matrix}...\end{matrix}` 命令， 而必须用 `\begin{array}{列样式}...\end{array}` 命令来创建。如：

```
\begin{array}{cc|c}
1 & 2 &3 \\
4 & 5 &6
\end{array}
```

的 `HTML` 输出为：

$$\begin{array}{cc|c} 1 & 2 & 3 \\ 4 & 5 & 6 \end{array}$$

5.11 分类表达式

`LATEX` 中，分类表达式用 `\begin{cases}...\end{cases}` 命令来创建，其中各行用 `\\` 分隔，用 `&` 指示要对齐的位置。如：

```
$$
f(n)= \begin{cases}
n/2, & \text{if $n$ is even} \\
3n+1, & \text{if $n$ is odd}
\end{cases}
$$
```

的 `HTML` 输出为：

$$f(n) = \begin{cases} n/2, & \text{if } n \text{ is even} \\ 3n+1, & \text{if } n \text{ is odd} \end{cases}$$

> cutemarked中需要输入 `\\\` ，则 `MathJax` 引擎接收到的字符是 `\\` ，因为cutemarked已经执行了一层转义

- `LATEX` 命令 `\begin{cases}...\end{cases}` （不需要 `$$...$$` ）也会生成一个区块公式。
- 也可以用表格或矩阵结合自适应大小的大括号来实现分类表达式。如:

  - 采用表格的实现：

    ```
    $$ f(n)= \left\\{
    \begin{array} {ll}
    n/2, & \text{if $n$ is even} \\\
    3n+1, & \text{if $n$ is odd}
    \end{array}
    \right.$$
    ```

    的 `HTML` 输出为：

    $$f(n) = \begin{cases} n/2, & \text{if } n \text{ is even} \\ 3n+1, & \text{if } n \text{ is odd} \end{cases}$$

  - 采用矩阵的实现：

    ```
    $$ f(n)= \left\\{
    \begin{matrix}
    n/2, & \text{if $n$ is even} \\\
    3n+1, & \text{if $n$ is odd}
    \end{matrix}
    \right.$$
    ```

    的 `HTML` 输出为：

    $$f(n) = \begin{cases} n/2, & \text{if } n \text{ is even} \\ 3n+1, & \text{if } n \text{ is odd} \end{cases}$$

- 若希望分类表达式的大括号在右侧，则必须采用表格或矩阵结合自适应大小的大括号来实现的方式。

5.12 公式标记与引用

在 `LATEX` 中，通过 `\tag{tagname}` 来标记公式，其中公式名字 `tagname` 由你自由取定。如果你还想在某个地方引用这个公式，你需要在 `\tag{tagname}` 命令之后通过 `\label{labelname}` 来设定一个 `labelname` ，然后在需要引用该公式的某个公式中使用 `\eqref{labelname}` 来引用这个公式。如标记公式：

```
$$
a :=x^{2}-y^{3} \tag{3.12} \label{eqx_y}
$$
```

的 `HTML` 输出为：

$$a := x^2 - y^3 \tag{3.12}$$

然后在某个公式中引用这个公式，如：

```
$$
a + y^{3} =x^{2} \eqref{eqx_y}\tag{3.13}
$$
```

的 `HTML` 输出为：

$$a + y^3 = x^2 \text{(3.12)} \tag{3.13}$$

- 在公式3.13中，可以看到它引用了公式3.12，通过超链接可以跳转到公式3.12
- 可以将 `\eqref{labelname}` 折叠起来，通过 `\stackrel` 命令。如：

  ```
  $$
  a + y^{3}\stackrel{\eqref{eqx_y}}{=}x^{2} \tag{3.13}
  $$
  ```

  的 `HTML` 输出为：

  $$a + y^3 \overset{(3.12)}{=} x^2 \tag{3.13}$$

5.13 定义公式颜色

LATEX 中公式默认都是黑色的文字，可以通过 \color{颜色值}{ ...} 来定义公式文字的颜色如，如：

```
$$
\color{red} { x^{2}+y^{2} }=z^{2}
$$
```

的 HTML 输出为：

$$\color{red} { x^{2}+y^{2} }=z^{2}$$

- 颜色值可以用 red,black,blue,white等表示，也可以用 #rgb 形式表示如 #FFF ，其中 r , g , b 分别为16进制数。

5.14 数学函数

LATEX 中的变量名是斜体，而函数名必须是竖直字体因此， LATEX 提供了下列命令：

```
\sin  \cos   \tan   \cot   \arcsin  \arccos  \arctan
\sec  \csc  \sinh  \cosh  \tanh  \coth
\exp  \ker  \limsup  \deg
\gcd  \lg  \ln  \log  \det  \hom  \arg
\dim  \lim  \inf  \liminf  \max  \min  \Pr  \sup
\bmod \pmod
```

它们在公式中的呈现为：

$$\sin \cos \tan \cot \arcsin \arccos \arctan$$
$$\sec \csc \sinh \cosh \tanh \coth$$
$$\exp \ker \limsup \deg$$
$$\gcd \lg \ln \log \det \hom \arg$$
$$\dim \lim \inf \liminf \max \min \Pr \sup$$

- \bmod \pmod 在公式中的呈现 \\(a \bmod b \quad a \pmod b\\) 的 HTML 输出为： $a \bmod b \quad a \pmod b$

5.15 数学运算符

- 四则运算：
  - 乘法 \times ，如 \\(x \times y \\) 的 HTML 输出为 $x \times y$
  - 除法 \div ，如 \\(x \div y \\) 的 HTML 输出为 $x \div y$
  - 正负号 \pm ，如 \\(\pm x \\) 的 HTML 输出为 $\pm x$
  - 倒置的正负号 \mp ，如 \\(\mp x \\) 的 HTML 输出为 $\mp x$
- 取模运算：有两个取模运算符号，其命令分别为 \bmod 与 \pmod 。区别在于 \pmod 会自动生成一对小括号。
  如 \\(a\bmod b \quad a\pmod b \\) 的 HTML 输出为：$a \bmod b \quad a \pmod b$
- 比较运算：
  - 小于、小于等于、大于、大于等于、不等于的命令依次为： \lt, \le, \gt, \ge, \eq ,\neq 。
    如 \\(x \lt y \quad x \le y \quad x \gt y \quad x \ge y \quad x \neq y\\) 的 HTML 输出为：
    $x \lt y \quad x \le y \quad x \gt y \quad x \ge y \quad x \neq y$

    > 等于命令直接用 = ，不存在 \eq 命令
    >
    > 小于等于也可以用 \leq 命令；大于等于也可以用 \geq 命令。如 \\(x \leq y \quad x \geq y \\) 的 HTML 输出为：$x \leq y \quad x \geq y$

  - 可以在上述比较运算符前面加上 \not ，表示不小于、不小于等于、不大于、不大于等于。
    如 \\(x \not\lt y \quad x \not\le y \quad x \not\gt y \quad x \not\ge y\\) 的 HTML 输出为：
    $x \not\lt y \quad x \not\le y \quad x \not\gt y \quad x \not\ge y$

    > 不存在 \not \neq 命令，虽然它可以渲染出一个≠符号，但是没有意义
    >
    > \not 命令不能单独使用

- 逻辑运算：有多个逻辑运算符号，其命令依次为

```
\land, \lor, \lnot, \forall, \exists
\because, \therefore, \not=, \not>, \not\subset
\top, \bot, \vdash, \vDash,
```

如

```
$$
\land\quad \lor\quad \lnot\quad \forall\quad \exists\\\
\because\quad \therefore\quad \not=\quad \not>\quad \not\subset\\\
\top\quad\bot\quad\vdash\quad\vDash\\\
$$
```

的 `HTML` 输出为

$$\land\quad \lor\quad \lnot\quad \forall\quad \exists\\\because\quad \therefore\quad \not=\quad \not>\quad \not\subset\\\top\quad\bot\quad\vdash\quad\vDash\\$$

- 求和 `\sum`，如 `\\( \sum_1^{n} \\)` 的 `HTML` 输出为 $\sum_1^n$
- 累乘 `\prod` 以及 `coprod`。
  - 如 `\\( \prod_1^{n} \\)` 的 `HTML` 输出为 $\prod_1^n$
  - 如 `\\( \coprod_1^{n} \\)` 的 `HTML` 输出为 $\coprod_1^n$
- 积分 `\int`，如 `\\( \int_1^{\infty} \\)` 的 `HTML` 输出为 $\int_1^\infty$
  - 双重积分 `\iint`，如 `\\( \iint_1^{\infty} \\)` 的 `HTML` 输出为 $\iint_1^\infty$
  - 三重积分 `\iiint`，如 `\\( \iiint_1^{\infty} \\)` 的 `HTML` 输出为 $\iiint_1^\infty$
  - 四重积分 `\iiiint`，如 `\\( \iiiint_1^{\infty} \\)` 的 `HTML` 输出为 $\iiiint_1^\infty$

    > 注意没有五重积分以及五重以上的积分

  - 曲线积分 `\oint`，如 `\\( \oint_D \\)` 的 `HTML` 输出为 $\oint_D$
- 极限 `\lim`，如 `\\( \lim_{n \rightarrow +\infty} \\)` 的 `HTML` 输出为 $\lim_{n\rightarrow+\infty}$
- 导数 `\prime`，如 `\\( f^{\prime}(x) \\)` 的 `HTML` 输出为 $f^{\prime}(x)$
  - 有多少阶导数就添加都少个 `\prime`，如三阶导数 `\\( f^{\prime\prime\prime}(x) \\)` 的 `HTML` 输出为 $f^{\prime\prime\prime}(x)$
  - 也可以用单引号来生成导数，如 `\\( f'(x) \\)` 的 `HTML` 输出为 $f'(x)$
- 集合运算：
  - 交集、并集、差集、子集、真子集、非真子集、父集的命令分别为：
    `\cup, \cap, \setminus, \subset, \subseteq, \subsetneq, \supset` 。如

    ```
    \\( A\cup B \quad  A \cap B \quad A \setminus B \quad  A \subset B
        \quad  A \subseteq B \quad A \subsetneq B \quad A \supset  B \quad\\)
    ```

    的 `HTML` 输出为 $A\cup B \quad A\cap B \quad A\setminus B \quad A\subset B \quad A\subseteq B \quad A\subsetneq B \quad A\supset B$
  - 属于、不属于命令分别为： `\in, \notin` 。如 `\\(x \in A \quad x \notin B \\)` 的 `HTML` 输出为： $x\in A \quad x\notin B$
  - 空集可以用 `\emptyset` 命令或者用 `\varnothing` 命令。 如 `\\(\emptyset \quad \varnothing \\)` 的 `HTML` 输出为：
    $\emptyset \quad \varnothing$
  - 还有另一种形式的交集、并集符号，分别为： `\vee, \wedge, \uplus, \sqcup` 。
    如 `\\(A \vee B \quad A \wedge B \quad A \uplus B \quad A \sqcup B \\)` 的 `HTML` 输出为 $A\vee B \quad A\wedge B \quad A\uplus B \quad A\sqcup B$
  - 存在大号版本的符号，分别为： `\bigcap, \bigcup, \bigvee, \bigwedge, \biguplus, \bigsqcup` 。如

    ```
    \\( A\bigcap B \quad  A\bigcup B\quad A\bigvee B\quad A\bigwedge B\quad
        A\biguplus B\quad A\bigsqcup B\quad\\)
    ```

    的输出为 $A\bigcap B \quad A\bigcup B \quad A\bigvee B \quad A\bigwedge B \quad A\biguplus B \quad A\bigsqcup B$
- 分式 `\frac{...}{...}`，如 `\\( \frac{a+1}{b+1} \\)` 的 `HTML` 输出为 $\frac{a+1}{b+1}$
  - 若分子为单个字符，则分子可以不用 `{}` ；若分母为单字符，则分母可以不用 `{}` 。 如 `\\( \frac ab \\)` 的 `HTML` 输出为 $\frac ab$
  - 也可以用 `\over` 命令来生成分式。 如 `\\(a+1 \over b+1\\)` 的 `HTML` 输出为 $\frac{a+1}{b+1}$

内建的 `\frac` 命令有两种显示形式：在区块中，它等效于 `\dfrac` ，显示较大；在行内，它等效于 `\tfrac` , 显示较小。而 `\dfrac` 类型的分式始终以较大的显示排版， `\tfrac` 类型的分式始终以较小的显示排版。 如行内分式： $\frac{a}{b}$ $\dfrac{a}{b}$ $\tfrac{a}{b}$ 区块分式：

$$\frac{a}{b} \quad \dfrac{a}{b} \quad \tfrac{a}{b}$$

- 根式 `\sqrt`，如 `\\( \sqrt a \\)` 的 `HTML` 输出为 $\sqrt a$

- 若需要指定幂次，则用 `\sqrt[n]` 命令，如 `\\( \sqrt[99] a \\)` 的 `HTML` 输出为 $\sqrt[99]{a}$
- 若不需要顶上的横线，则用 `\surd` 命令，如 `\\( \surd a \\)` 的 `HTML` 输出为 $\surd a$

- 排列 `\choose`，如 `\\( n+1 \choose 2k \\)` 的 `HTML` 输出为 $\binom{n+1}{2k}$
  - 排列还有一个命令 `\binom{...}{...}`，如 `\\(\binom {n+1}{2k} \\)` 的 `HTML` 输出为 $\binom{n+1}{2k}$。其中若某个分组为单字符，则不用 `{}`。 如 `\\(\binom nk \\)` 的 `HTML` 输出为$\binom{n}{k}$。

5.16 数学符号

- 星号：有多种星号，其命令依次为

```
\star, \ast, \oplus, \circ,
\bullet, \bigotimes, \bigoplus,
```

如

```
$$
\star\quad \ast\quad \oplus\quad \circ\quad
\bullet\quad \bigotimes\quad \bigoplus
$$
```

的 `HTML` 输出为

$$\star \quad * \quad \oplus \quad \circ \quad \bullet \quad \bigotimes \quad \bigoplus$$

- 箭头：有多种箭头，其命令依次为

```
\uparrow, \downarrow, \leftarrow, \rightarrow,
\Uparrow, \Downarrow, \Leftarrow, \Rightarrow,
\longrightarrow, \longRightarrow, \Longleftarrow, \Longrightarrow,
\to, \mapsto
```

如

```
$$
\uparrow \quad \downarrow\quad \leftarrow\quad \rightarrow\\\
\Uparrow\quad \Downarrow\quad \Leftarrow\quad \Rightarrow\\\
\longleftarrow\quad\longrightarrow\quad\Longleftarrow\quad\Longrightarrow\\\
\to\quad \mapsto
$$
```

的 `HTML` 输出为

$$\uparrow \quad \downarrow \quad \leftarrow \quad \rightarrow$$
$$\Uparrow \quad \Downarrow \quad \Leftarrow \quad \Rightarrow$$
$$\longleftarrow \quad \longrightarrow \quad \Longleftarrow \quad \Longrightarrow$$
$$\to \quad \mapsto$$

- 约等于:有多种约等于符号，其命令依次为：

```
\approx, \sim, \cong, \equiv,\prec
```

如

```
$$
\approx\quad \sim\quad \cong\quad \equiv \quad \prec\
$$
```

的 `HTML` 输出为

$$\approx \quad \sim \quad \cong \quad \equiv \quad \prec$$

- 无穷大命令为 `\infty`，如 `\\( \infty \\)` 的 `HTML` 输出为 ∞
- 偏导数符号为 `\partial`，如 `\\( \frac {\partial ^{2}f}{\partial x^{2}}\\)` 的 `HTML` 输出为 $\frac{\partial^2 f}{\partial x^2}$
- `\nabla` 为倒三角，如 `\\( \nabla \\)` 的 `HTML` 输出为 ∇
- 省略号有两种，分别是 `\ldots` 命令以及 `\cdots` 命令。其中 `\ldots` 生成的省略号位置偏低，下对齐； `\lcdots` 生成的省略号位

置居中，居中对齐。 如 `\\( 1\ldots n \quad 1\cdots n \\)` 生成的 `HTML` 输出为$1\ldots n \quad 1\cdots n$

- 顶部符号和底部符号
  - 对于单字符的顶部符号和底部符号，只能用下列命令：

```
\hat, \vec,  \dot, \ddot,
\check, \breve, \bar
```

如

```
$$
\hat x\quad\vec x\quad\dot x\quad\ddot x\quad\\\
\check x\quad\breve x\quad\bar x
$$
```

的 `HTML` 输出为

$$\hat{x} \quad \vec{x} \quad \dot{x} \quad \ddot{x}$$
$$\check{x} \quad \breve{x} \quad \bar{x}$$

  - 对于多字符的顶部符号和底部符号，需要用下列命令

```
\widehat, \overline,\overrightarrow,
\underline, \overbrace, \underbrace
```

如

```
$$
\widehat {ABC}\quad \overline {ABC}\quad\overrightarrow {ABC}\quad \\\
\underline {ABC}\quad\overbrace {ABC}\quad\underbrace {ABC}\quad
$$
```

的 `HTML` 输出为

$$\widehat{ABC} \quad \overline{ABC} \quad \overrightarrow{ABC}$$
$$\underline{ABC} \quad \overbrace{ABC} \quad \underbrace{ABC}$$

> 若对多字符使用单字符的顶部符号和底部符号，则该符号只会占据单个字符， 如 `\\( \bar{ABC}\\)` 的 `HTML` 输出为$\bar{ABC}$

5.18 注意事项

- 当在指数或者积分中需要分数时，不要使用 `\frac` 命令，而是要用一个水平的 `/` 符号代替。 如: $e^{i\frac{\pi}{2}}\text{(bad)}$ $e^{i\pi/2}\text{(good)}$
- 当用 `|` 做分隔符来进行排版布局时，你可能需要的是 `\mid` 命令。如：

$$\{x|\sqrt{x}\in\mathbb{Z}\}:\text{seperate with |}$$
$$\{x\mid\sqrt{x}\in\mathbb{Z}\}:\text{seperate with \mid}$$

- 多重积分不要使用 `\int\int` ，必须使用 `\iint` 等特殊形式
- 多重积分中，在微分之间使用 `\,` 保留间隙。如：

$$\iint_D f(x,y)dxdy:\text{no seperate}$$
$$\iint_D f(x,y)dx\,dy:\text{seperate with \,}$$

- 书写连分数时，不能直接用 `\frac` 以及 `\over` ，而必须用 `\cfrac` ，因为前者会导致分母的显示越来越小。如：

$$a_0+\frac{1}{a_1+\frac{2}{a_2+\frac{3}{a_3+\cdots}}}:\text{use \frac}$$
$$a_0+\cfrac{1}{a_1+\cfrac{2}{a_2+\cfrac{3}{a_3+\cdots}}}:\text{use \cfrac}$$

6 cutemarked流程图扩展

cutemarked的流程图扩展使用了 `mermaid` 开源框架，其使用方法为:

```
~~~mermaid
//这里插入mermaid语法
~~~
```

下面是mermaid语法

6.1 图形

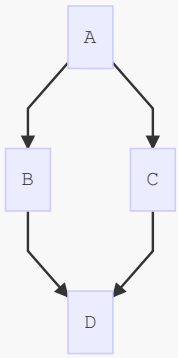
声明一个图形用 `graph` 命令，分别有 `graph TB` ， `graph BT` ， `graph RL` ， `graph LR` ， `graph TD` 五种类型。

6.1.1 `graph TB`

`graph TB` :从top到bottom，如：

```
~~~mermaid
graph TB;
A-->B;
A-->C;
B-->D;
C-->D;
~~~
```

的图形为：

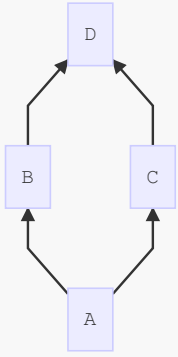


6.1.2 `graph BT`

`graph BT` :从bottom到top，如：

```
~~~mermaid
graph BT;
A-->B;
A-->C;
B-->D;
C-->D;
~~~
```

的图形为：

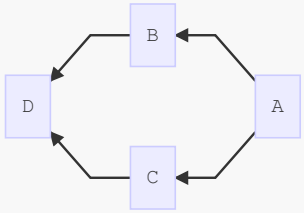


6.1.3 `graph RL`

`graph RL` :从right到left，如：

```
~~~mermaid
graph RL;
A-->B;
A-->C;
B-->D;
C-->D;
~~~
```

的图形为：

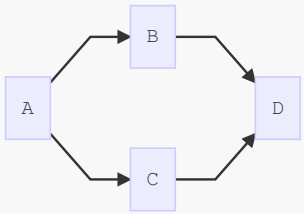


6.1.4 `graph LR`

`graph LR`:从left到right，如：

```
~~~mermaid
graph LR;
A-->B;
A-->C;
B-->D;
C-->D;
~~~
```

的图形为：

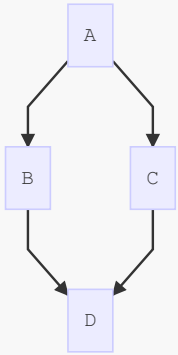


6.1.5 `graph TD`

`graph TD`:等同于TB，如：

```
~~~mermaid
graph TD;
A-->B;
A-->C;
B-->D;
C-->D;
~~~
```

的图形为：



6.2 节点

6.2.1 默认节点

默认节点用 `id` 表示，如：

```
~~~mermaid
graph LR
id1
~~~
```

的图形为：

id1

6.2.2 带文本的节点

可以设置节点方框中的文本替代默认的 `id` 名字，如：

```
~~~mermaid
graph LR
id1[This is the text in the box]
~~~
```

的图形为：

This is the text in the box

> `id` 与 `[` 之间必须没有空隔

- 你可以对某个 `id` 的方框多次设置文本，以最近一次设置的文本为主，如：

  ```
  ~~~mermaid
  graph LR
  id1[This is the text in the box]
  id1[This is the text in the box2]
  ~~~
  ```

  的图形为：

  This is the text in the box2

6.2.3 圆角节点

默认的节点是矩形框，你可以通过 `id( )` 设置成圆角节点，如：

```
~~~mermaid
graph LR
id1(This is the text in the box)
~~~
```

的图形为：

> `id` 与 `(` 之间必须没有空隔

6.2.4 圆形节点

默认的节点是矩形框，你可以通过 `id(( ))` 设置成圆形节点，如：

```
~~~mermaid
graph LR
id1((This is the text in the circle))
~~~
```

的图形为：

> `id` 与 `((` 之间必须没有空隔

6.2.5 非对称节点

默认的节点是矩形框，你可以通过 `id> ]` 设置成非对称节点，如：

```
~~~mermaid
graph LR
id1>This is the text in the box]
~~~
```

的图形为：

> `id` 与 `>` 之间必须没有空隔

6.2.6 菱形节点

默认的节点是矩形框，你可以通过 `id{ }` 设置成菱形节点，如：

```
~~~mermaid
graph LR
id1{This is the text in the box}
~~~
```

的图形为：

> `id` 与 `{` 之间必须没有空隔

6.3 节点的边

边有多种形式，同时也可以设置边的文字

6.3.1 实线边

6.3.1.1 带箭头无文字的实线边

带箭头无文字的实线边语法为：

```
~~~mermaid
graph LR
A-->B
~~~
```

的图形为：

> 可以用 `graph LR` ，也可以用 `graph LR;`

6.3.1.2 不带箭头和文字的实线边

不带箭头和文字的实线边语法为：

```
~~~mermaid
graph LR
A---B
~~~
```

的图形为：

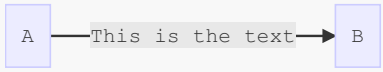
> 可以用`graph LR`，也可以用`graph LR;`

6.3.1.3 带箭头有文字的实线边

带箭头和文字的实线边可以有两种语法,其中

```
    ~~~mermaid
    graph LR;
    A-- text -->B
    ~~~
```

的图形为：

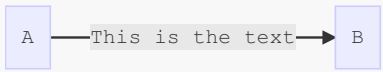


> 可以用`graph LR`，也可以用`graph LR;`

另一种形式：

```
    ~~~mermaid
    graph LR
    A-->|This is the text|B
    ~~~
```

的图形为：



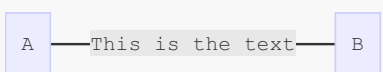
> 可以用`graph LR`，也可以用`graph LR;`

6.3.1.4 有文字无箭头的实线边

有文字无箭头的实线边可以有两种语法,其中

```
    ~~~mermaid
    graph LR
    A-- This is the text --- B
    ~~~
```

的图形为：

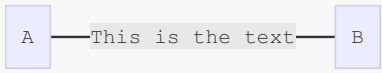


> 注意`A`后面跟的是两横`--`，`B`前面是三横`---`
>
> 可以用`graph LR`，也可以用`graph LR;`

另一种形式：

```
~~~mermaid
graph LR;
A---|This is the text|B
~~~
```

的图形为：



> 注意`A`后面跟的是三横一竖`---|`，`B`前面是一竖`|`
>
> 可以用`graph LR`，也可以用`graph LR;`

6.3.2 虚线边

6.3.2.1 带箭头不带文字

带箭头无文字虚线边语法为

```
~~~mermaid
graph LR
A-.->B
~~~
```

的图形为：

> 可以用`graph LR`，也可以用`graph LR;`

6.3.2.2 不带箭头和文字

不带箭头和文字虚线边语法为

```
~~~mermaid
graph LR
A-.-B
~~~
```

的图形为：

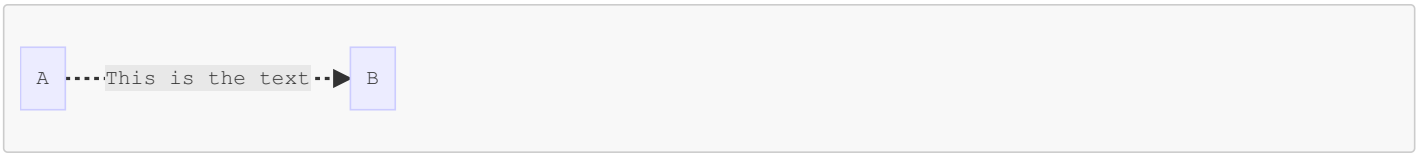
> 可以用`graph LR`，也可以用`graph LR;`

6.3.2.3 带箭头和文字

带箭头和文字虚线边语法为

```
~~~mermaid
graph LR
A-. This is the text .->B
~~~
```

的图形为：



> 可以用`graph LR`，也可以用`graph LR;`

6.3.2.4 带文字不带箭头

带文字不带箭头虚线边语法为

```
    ~~~mermaid
    graph LR
    A-. This is the text .-B
    ~~~
```

的图形为：

> 可以用`graph LR`，也可以用`graph LR;`

6.3.3 粗实线边

6.3.3.1 带箭头无文字

带箭头无文字的粗实线边语法为：

```
    ~~~mermaid
    graph LR
    A==>B
    ~~~
```

的图形为：

> 可以用`graph LR`，也可以用`graph LR;`

6.3.3.2 不带箭头和文字

不带箭头和文字的粗实线边语法为：

```
    ~~~mermaid
    graph LR
    A===B
    ~~~
```

的图形为：

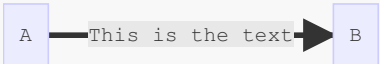
> 可以用`graph LR`，也可以用`graph LR;`

6.3.3.3 带箭头有文字

带箭头和文字的粗实线边可以有两种语法,其中

```
~~~mermaid
graph LR;
A== This is the text ==>B
~~~
```

的图形为:

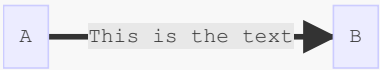


> 可以用`graph LR`，也可以用`graph LR;`

另一种形式:

```
~~~mermaid
graph LR
A==>|This is the text|B
~~~
```

的图形为:



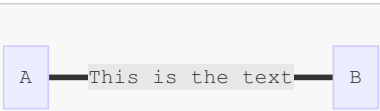
> 可以用`graph LR`，也可以用`graph LR;`

6.3.3.4 有文字无箭头

有文字无箭头的粗实线边可以有两种语法,其中

```
~~~mermaid
graph LR
A== This is the text === B
~~~
```

的图形为:

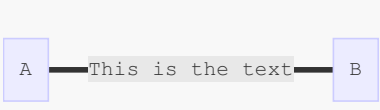


> 注意`A`后面跟的是两个等号`==`，`B`前面是三个等号`===`
>
> 可以用`graph LR`，也可以用`graph LR;`

另一种形式:

```
~~~mermaid
graph LR;
A===|This is the text|B
~~~
```

的图形为:



> 注意`A`后面跟的是三个等号一竖`===|`，`B`前面是一竖`|`

> 可以用`graph LR`，也可以用`graph LR;`

6.4 特殊字符

为了在文字中使用特殊字符，你必须将文字用引号括起来，如想在文字中使用`()`，则比如：

```
~~~mermaid
graph LR
id1["This is the (text) in the box"]
~~~
```

你可以在引号中使用特殊字符的entity code，如：

```
~~~mermaid
graph LR
A["A double quote:#quot;"] -->B["A dec char:#9829;"]
~~~
```

> 在`cutemarked`中，这两种语法都无法正确解析，原因未知

6.5 子图

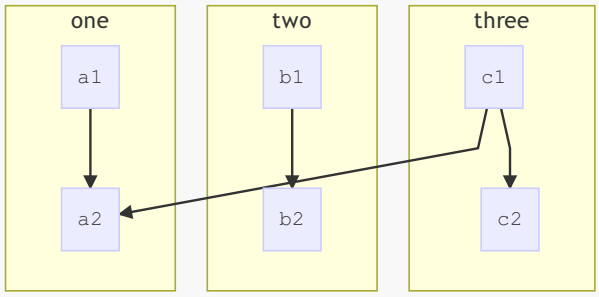
可以通过`subgraph`命令建立子图，其语法为：

```
subgraph title
    graph definition
end
```

如：

```
~~~mermaid
graph TB
    subgraph one
    a1-->a2
    end
    subgraph two
    b1-->b2
    end
    subgraph three
    c1-->c2
    end
    c1-->a2
~~~
```

的图形为：



6.6 节点和边的样式

6.6.1节点的样式

可以通过`style`命令设置节点的样式，其语法为：

```
    style id fill:#f9f,stroke:#333,stroke-width:2px,stroke-dasharray: 5, 5;
```

- `fill`为节点的填充颜色，`stroke`为节点的边框颜色，`stroke-width`为边框的线宽，，`stroke-dasharray`为边框的虚线样式（如果没有此选项则为实线）

如：

```
    ~~~mermaid
    graph LR
        id1(Start)-->id2(Stop);
        A-->B
        style id1 fill:#f9f,stroke:#333,stroke-width:4px;
        style id2 fill:#ccf,stroke:#f66,stroke-width:2px,stroke-dasharray: 5, 5;
    ~~~
```

的图形为：

6.6.2边的样式

可以通过`linkStyle`设置边的样式，其语法为：

```
    linkStyle 0 stroke:#ff3,stroke-width:4px;
```

- `0`表示设置图形的第0条边（从0开始计数），`stroke`为边的颜色，`stroke-width`为边的线宽，，`stroke-dasharray`为边框的虚线样式（如果没有此选项则为实线）

> 由于边的定义时没有`边id`这一说法，因此引用边时采用了边的序号，其中序号按照图形中边的定义顺序依次给出。

如：

```
    ~~~mermaid
    graph LR
            id1(Start)-->id2(Stop);
            linkStyle 0 stroke:#ff3,stroke-width:4px,stroke-dasharray: 5, 5;
    ~~~
```

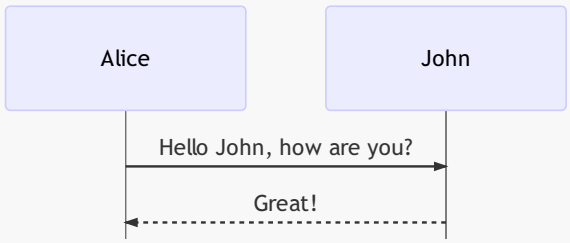
的图形为：

6.7 时序图

可以通过`sequenceDiagram`命令创建时序图，如：

```
    ~~~mermaid
    sequenceDiagram
        Alice->>John: Hello John, how are you?
        John-->>Alice: Great!
    ~~~
```

的图形为：



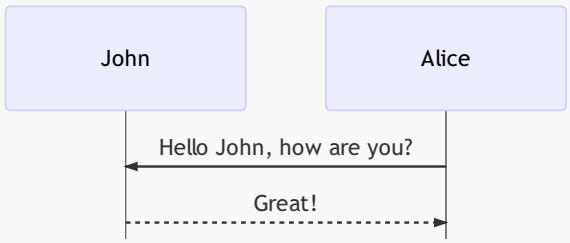
下面是时序图的语法

6.7.1 定义参与者

参与者可以采取类似上述例子中的隐式定义的方法定义，所有的参与者的展示顺序与message中参与者的出现顺序一致；你也可以显式的定义参与者，此时其展示顺序与定义顺序一致。如：

```
~~~mermaid
sequenceDiagram
    participant John
    participant Alice
    Alice->>John: Hello John, how are you?
    John-->>Alice: Great!
~~~
```

的图形为：



6.7.2 message

定义message的语法为：

```
[Actor1][Arrow][Actor2]:message text
```

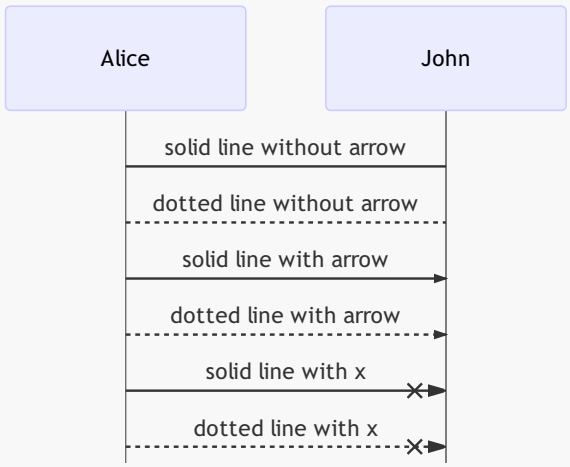
其中 `[Actor1]` 是消息的发送者， `[Actor2]` 是消息的接收者， `[Arrow]` 是线条类型。目前 `mermaid` 支持下列六种线条：

- `->` 为无箭头实线
- `-->` 为无箭头点线
- `->>` 为带箭头实线
- `-->>` 为带箭头点线
- `-x` 为实线,端点为 `x`
- `--` 为点线,端点为 `x`

如：

```
~~~mermaid
sequenceDiagram
    Alice->John: solid line without arrow
    Alice-->John: dotted line without arrow
    Alice->>John: solid line with arrow
    Alice-->>John: dotted line with arrow
    Alice-xJohn: solid line with x
    Alice--xJohn: dotted line with x
~~~
```

的图形为：



6.7.3 注释

定义注释的语法为：

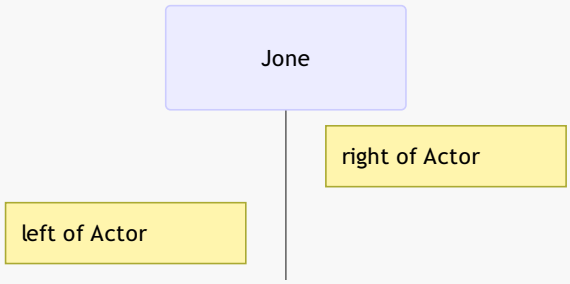
```
    Note [right of | left of ] [Actor]: note text
```

其中`[Actor]`是被注释的参与者，`[right of | left of]`是该注释相对于参与者的位置。

如：

```
    ~~~mermaid
    sequenceDiagram
        participant Jone
        Note right of Jone: right of Actor
        Note left of Jone: left of Actor
    ~~~
```

的图形为：



6.7.4 Loop

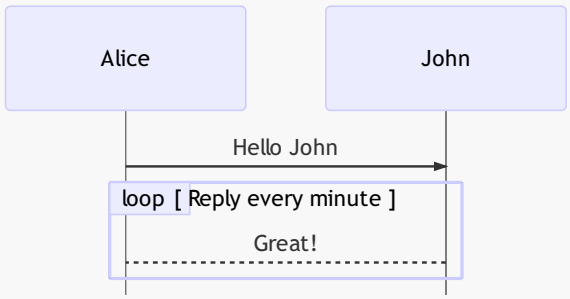
可以通过`loop`命令创建Loop，其格式为：

```
    loop Loop_text
    ... statements ...
    end
```

如：

```
    ~~~mermaid
    sequenceDiagram
        Alice->John: Hello John
            loop Reply every minute
        John-->Alice: Great!
        end
    ~~~
```

的图形为：



6.7.5 可选路径

可以通过 `alt` 命令创建可选路径，其格式为：

```
    alt Describing_text
    ... statements ...
    else
    ... statements ...
    end
```

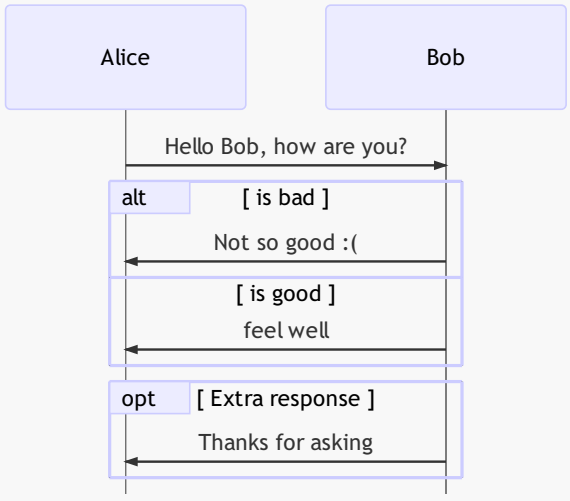
若使用 `opt` 命令则也可以不需要 `else` ，：

```
    opt Describing_text
    ... statements ...
    end
```

如：

```
    ~~~mermaid
    sequenceDiagram
        Alice->>Bob: Hello Bob, how are you?
        alt is sick
            Bob->>Alice: Not so good :(
        else is well
            Bob->>Alice: Feeling fresh like a daisy
        end
        opt Extra response
            Bob->>Alice: Thanks for asking
        end
    ~~~
```

的图形为：



6.7.6 样式

时序图的样式由.css文件的class定义，用到的类型有：

```
Class   Description
actor   Style for the actor box at the top of the diagram.
text.actor  Styles for text in the actor box at the top of the diagram.
actor-line  The vertical line for an actor.
messageLine0    Styles for the solid message line.
messageLine1    Styles for the dotted message line.
messageText     Defines styles for the text on the message arrows.
labelBox    Defines styles label to left in a loop.
labelText   Styles for the text in label for loops.
loopText    Styles for the text in the loop box.
loopLine    Defines styles for the lines in the loop box.
note    Styles for the note box.
noteText    Styles for the text on in the note boxes.
```

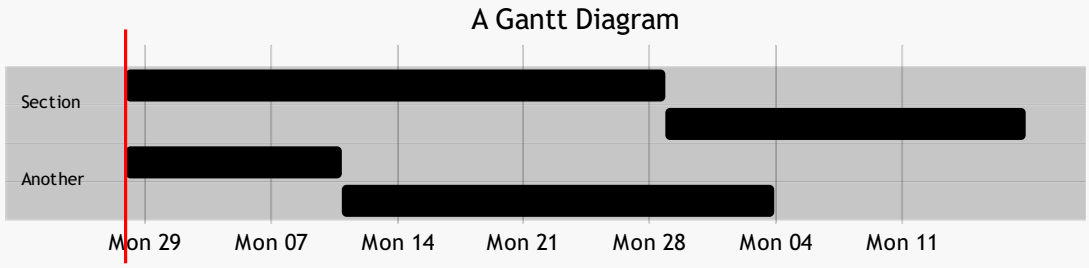
6.8 Gant图

Gant图用于说明工程进度，它通过指定元素的起止时间来说明工程进度信息。

可以通过 `gantt` 命令创建时序图，如：

```
    ~~~mermaid
    gantt
        title A Gantt Diagram

    section Section
    A task           :a1, 2014-01-01, 30d
    Another task     :after a1  , 20d
    section Another
    Task in sec      :2014-01-12  , 12d
    anther task      : 24d
    ~~~
```

的图形为：

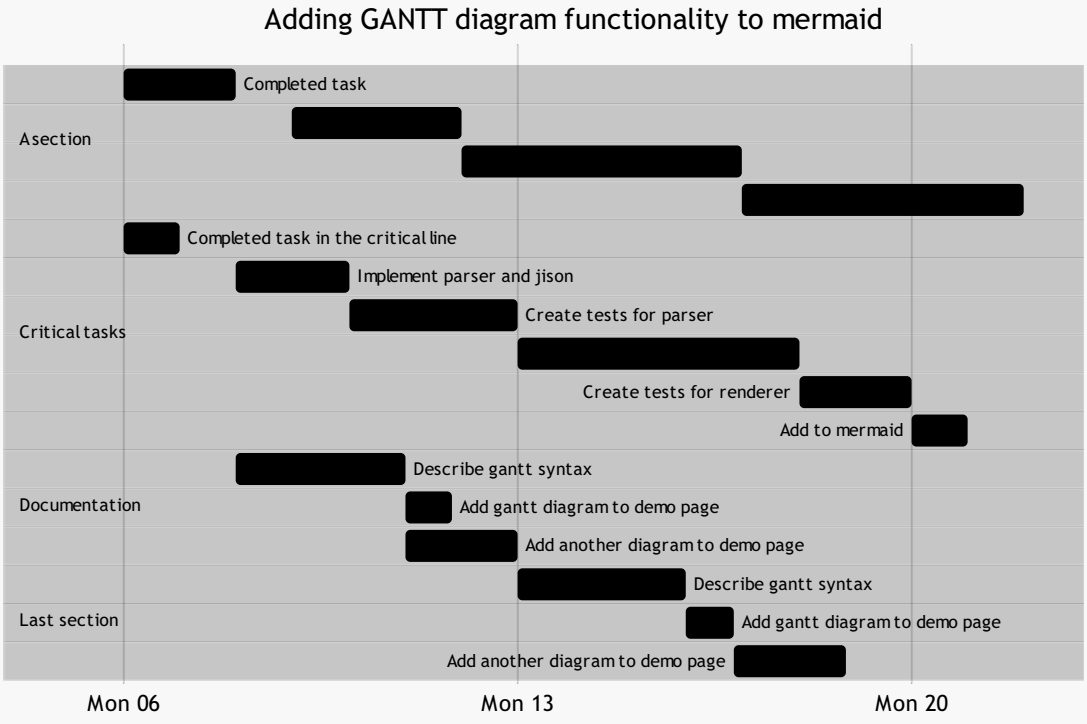


6.8.1 语法

其语法参考下面的示例：

```
~~~mermaid
    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
~~~
```

的图形为：



- `title` 为标题元素
- `section` 为段元素
    - `section` 中每条 `task` 的组成为 `task description: 任务属性, 任务name, 任务位置, 任务开始时间, 任务结束时间`

6.8.2 样式

Gant图的样式由.css的class定义，用到的类型有：

?官方文档有误

7 LaTeX 所有符号表

7.1 重音符

$\hat{a}$: \hat{a}	$\check{a}$: \check{a}	$\tilde{a}$: \tilde{a}
$\grave{a}$: \grave{a}	$\dot{a}$: \dot{a}	$\ddot{a}$: \ddot{a}
$\bar{a}$: \bar{a}	$\vec{a}$: \vec{a}	$\widehat{AAA}$: \widehat{AAA}
$\acute{a}$: \acute{a}	$\breve{a}$: \breve{a}	$\widetilde{AAA}$: \widetilde{AAA}
$\mathring{a}$: \mathring{a}		

7.2 希腊字母

α : \alpha	θ : \theta	o : o	υ : \upsilon
β : \beta	ϑ : \vartheta	π : \pi	ϕ : \phi
γ : \gamma	ι : \iota	ϖ : \varpi	φ : \varphi
δ : \delta	κ : \kappa	ρ : \rho	χ : \chi
ϵ : \epsilon	λ : \lambda	ϱ : \varrho	ψ : \psi
ε : \varepsilon	μ : \mu	σ : sigma	ω : \omega
ζ : \zeta	ν : \nu	ς : \varsigma	
η : \eta	ξ : \xi	τ : \tau	
Γ : \Gamma	Λ : \Lambda	Σ : \Sigma	Ψ : \Psi
Δ : \Delta	Ξ : \Xi	Υ : \Upsilon	Ω : \Omega
Θ : \Theta	Π : \Pi	Φ : \Phi	

7.3 二元关系

$<$: <	$>$: >	$=$: =
$<$: \lt	$>$: \gt	
$\leq$: \leq	$\geq$: \geq	$\equiv$: \equiv
$\leq$: \le	$\geq$: \ge	
$\ll$: \ll	$\gg$: \gg	$\doteq$: \doteq
$\prec$: \prec	$\succ$: \succ	$\sim$: \sim
$\preceq$: \preceq	$\succeq$: \succeq	$\simeq$: \simeq
$\subset$: \subset	$\supset$: \supset	$\approx$: \approx
$\subseteq$: \subseteq	$\supseteq$: \supseteq	$\cong$: \cong
$\sqsubset$: \sqsubset	$\sqsupset$: \sqsupset	$\bowtie$: \Join
$\sqsubseteq$: \sqsubseteq	$\sqsupseteq$: \sqsupseteq	$\bowtie$: \bowtie
$\in$: \in	$\ni$: \ni	$\propto$: \propto
	$\ni$: \owns	
$\vdash$: \vdash	$\dashv$: \dashv	$\models$: \models
$\mid$: \mid	$\parallel$: \parallel	$\perp$: \perp
$\smile$: \smile	$\frown$: \frown	$\asymp$: \asymp
$:$: :	$\notin$: \notin	$\neq$: \neq
		$\neq$: \ne

7.4 二元运算符

$+$: +	$-$: -	
$\pm$: \pm	$\mp$: \mp	$\triangleleft$: \triangleleft
$\cdot$: \cdot	$\div$: \div	$\triangleright$: \triangleright
$\times$: \times	$\setminus$: \setminus	$\star$: \star
$\cup$: \cup	$\cap$: \cap	$*$: \ast
$\sqcup$: \sqcup	$\sqcap$: \sqcap	$\circ$: \circ
$\vee$: \vee	$\wedge$: \wedge	$\bullet$: \bullet
$\vee$: \lor	$\wedge$: \land	
$\oplus$: \oplus	$\ominus$: \ominus	$\diamond$: \diamond
$\odot$: \odot	$\oslash$: \oslash	$\uplus$: \uplus
$\otimes$: \otimes	$\bigcirc$: \bigcirc	$\amalg$: \amalg
$\bigtriangleup$: \bigtriangleup	$\bigtriangledown$: \bigtriangledown	$\dagger$: \dagger
$\lhd$: \lhd	$\rhd$: \rhd	$\ddagger$: \ddagger
$\unlhd$: \unlhd	$\unrhd$: \unrhd	$\wr$: \wr

7.5 大号操作符

$\sum$: \sum	$\bigcup$: \bigcup	$\bigvee$: \bigvee
$\prod$: \prod	$\bigcap$: \bigcap	$\bigwedge$: \bigwedge
$\coprod$: \coprod	$\bigsqcup$: \bigsqcup	$\biguplus$: \biguplus
$\int$: \int	$\oint$: \oint	$\bigodot$: \bigodot
$\bigoplus$: \bigoplus	$\bigotimes$: \bigotimes	

7.6 箭头

$\leftarrow$: \leftarrow $\longleftarrow$: \longleftarrow
$\leftarrow$: \gets
$\rightarrow$: \rightarrow $\longrightarrow$: \longrightarrow
$\rightarrow$: \to
$\leftrightarrow$: \leftrightarrow $\longleftrightarrow$: \longleftrightarrow
$\Leftarrow$: \Leftarrow $\Longleftarrow$: \Longleftarrow
$\Rightarrow$: \Rightarrow $\Longrightarrow$: \Longrightarrow
$\Leftrightarrow$: \Leftrightarrow $\Longleftrightarrow$: \Longleftrightarrow
$\mapsto$:: \mapsto $\longmapsto$: \longmapsto
$\hookleftarrow$:: \hookleftarrow $\hookrightarrow$:: \hookrightarrow
$\leftharpoonup$:: \leftharpoonup $\rightharpoonup$: \rightharpoonup
$\leftharpoondown$: \leftharpoondown $\rightharpoondown$: \rightharpoondown
$\rightleftharpoons$: \rightleftharpoons $\iff$: \iff
$\uparrow$: \uparrow $\downarrow$: \downarrow
$\updownarrow$: \updownarrow $\Uparrow$: \Uparrow
$\Downarrow$: \Downarrow $\Updownarrow$: \Updownarrow
$\nearrow$: \nearrow $\searrow$: \searrow
$\swarrow$: \swarrow $\nwarrow$: \nwarrow
$\leadsto$: \leadsto

7.7 重音箭头

$\overrightarrow{AB}$: \overrightarrow{AB} $\underrightarrow{AB}$: \underrightarrow{AB}
$\overleftarrow{AB}$: \overleftarrow{AB} $\underleftarrow{AB}$: \underleftarrow{AB}
$\overleftrightarrow{AB}$: \overleftrightarrow{AB} $\underleftrightarrow{AB}$: \underleftrightarrow{AB}

7.8 分隔符

(: ( ) :) $\uparrow$: \uparrow
[: [ ] :] $\downarrow$: \downarrow
[: \lbrack] : \rbrack
$\{$: \\{ $\}$: \\} $\updownarrow$: \updownarrow
$\{$: \lbrace $\}$: \rbrace
< : < > : > $\Uparrow$: \Uparrow
$\langle$ :: \langle $\rangle$: \rangle
| : | $\|$: \| $\Downarrow$: \Downarrow
| :: \vert $\|$: \Vert
/ : / \ : \\backslash $\Updownarrow$: \Updownarrow
$\lfloor$ :: \lfloor $\rfloor$: \rfloor
$\rceil$ :: \rceil $\lceil$: \lceil

7.9 大号分隔符

$\lgroup$: \lgroup $\rgroup$: \rgroup $\lmoustache$: \lmoustache
$\arrowvert$: \arrowvert $\Arrowvert$: \Arrowvert $\bracevert$: \bracevert
$\rmoustache$: \rmoustache

7.10 各式各样的符号

$\dots$: \dots $\cdots$: \cdots $\vdots$: \vdots $\ddots$: \ddots
$\hbar$: \hbar $\imath$: \imath $\jmath$: \jmath ℓ : \ell
$\Re$: \Re $\Im$: \Im $\aleph$: \aleph $\wp$: \wp
$\forall$: \forall $\exists$: \exists $\mho$: \mho ∂ : \partial
' : ' $\prime$: \prime $\emptyset$: \emptyset ∞ : \infty
∇ : \nabla $\triangle$: \triangle $\Box$: \Box $\Diamond$: \Diamond
$\bot$: \bot $\top$: \top $\angle$: \angle $\surd$: \surd
$\diamondsuit$: \diamondsuit $\heartsuit$: \heartsuit $\clubsuit$: \clubsuit $\spadesuit$: \spadesuit
$\neg$: \neg $\flat$: \flat $\natural$: \natural $\sharp$: \sharp
$\neg$: \lnot

7.11 非数学符号

§ : \S

7.12 AMS分隔符

$\ulcorner$: \ulcorner $\urcorner$: \urcorner $\llcorner$: \llcorner $\lrcorner$: \lrcorner
$\lvert$: \lvert $\rvert$: \rvert $\lVert$: \lVert $\rVert$: \rVert

7.13 希腊和希伯来语

$\digamma$: \digamma $\varkappa$: \varkappa $\beth$: \beth $\gimel$: \gimel $\daleth$: \daleth

7.14 数学字体

ABCDEabcde1234 : \mathrm{ABCDE abcde 1234}
ABCDEabcde1234 : \mathit{ABCDE abcde 1234}
$\mathcal{ABCDEabcde1234}$: \mathcal{ABCDE abcde 1234}
$\mathscr{ABCDEabcde1234}$: \mathscr{ABCDE abcde 1234}
$\mathfrak{ABCDEabcde1234}$: \mathfrak{ABCDE abcde 1234}
$\mathbb{ABCDEabcde1234}$: \mathbb{ABCDE abcde 1234}

7.15 AMS 二元操作符

$\dotplus$: \dotplus	$\centerdot$: \centerdot	
$\ltimes$: \ltimes	$\rtimes$: \rtimes	$\divideontimes$: \divideontimes
$\Cup$: \doublecup	$\Cap$: \doublecap	$\smallsetminus$: \smallsetminus
$\veebar$: \veebar	$\barwedge$: \barwedge	$\doublebarwedge$: \doublebarwedge
$\boxplus$: \boxplus	$\boxminus$: \boxminus	$\circleddash$: \circleddash
$\boxtimes$: \boxtimes	$\boxdot$: \boxdot	$\circledcirc$: \circledcirc
$\intercal$: \intercal	$\circledast$: \circledast	$\rightthreetimes$: \rightthreetimes
$\curlyvee$: \curlyvee	$\curlywedge$: \curlywedge	$\leftthreetimes$: \leftthreetimes

7.16 AMS 二元关系

$\lessdot$: \lessdot	$\gtrdot$: \gtrdot	$\doteqdot$: \doteqdot
$\leqslant$: \leqslant	$\geqslant$: \geqslant	$\risingdotseq$: \risingdotseq
$\eqslantless$: \eqslantless	$\eqslantgtr$: \eqslantgtr	$\fallingdotseq$: \fallingdotseq
$\leqq$: \leqq	$\geqq$: \geqq	$\eqcirc$: \eqcirc
$\lll$: \lll	$\ggg$: \ggg	$\circeq$: \circeq
$\llless$: \llless		
$\lesssim$: \lesssim	$\gtrsim$: \gtrsim	$\triangleq$: \triangleq
$\lessapprox$: \lessapprox	$\gtrapprox$: \gtrapprox	$\bumpeq$: \bumpeq
$\lessgtr$: \lessgtr	$\gtrless$: \gtrless	$\Bumpeq$: \Bumpeq
$\lesseqgtr$: \lesseqgtr	$\gtreqless$: \gtreqless	$\thicksim$: \thicksim
$\lesseqqgtr$: \lesseqqgtr	$\gtreqqless$: \gtreqqless	$\thickapprox$: \thickapprox
$\preccurlyeq$: \preccurlyeq	$\succcurlyeq$: \succcurlyeq	$\approxeq$: \approxeq
$\curlyeqprec$: \curlyeqprec	$\curlyeqsucc$: \curlyeqsucc	$\backsim$: \backsim
$\precsim$: \precsim	$\succsim$: \succsim	$\backsimeq$: \backsimeq
$\precapprox$: \precapprox	$\succapprox$: \succapprox	$\vDash$: \vDash
$\subseteqq$: \subseteqq	$\supseteqq$: \supseteqq	$\Vdash$: \Vdash
$\shortparallel$: \shortparallel	$\Supset$: \Supset	$\Vvdash$: \Vvdash
$\blacktriangleleft$: \blacktriangleleft	$\sqsupset$: \sqsupset	$\backepsilon$: \backepsilon
$\vartriangleright$: \vartriangleright	$\because$: \because	$\varpropto$: \varpropto
$\blacktriangleright$: \blacktriangleright	$\Subset$: \Subset	$\between$: \between
$\trianglerighteq$: \trianglerighteq	$\smallfrown$: \smallfrown	$\pitchfork$: \pitchfork
$\vartriangleleft$: \vartriangleleft	$\shortmid$: \shortmid	$\smallsmile$: \smallsmile
$\trianglelefteq$: \trianglelefteq	$\therefore$: \therefore	$\sqsubset$: \sqsubset

7.17 AMS箭头

$\dashleftarrow$: \dashleftarrow	$\dashrightarrow$: \dashrightarrow
$\leftleftarrows$: \leftleftarrows	$\rightrightarrows$: \rightrightarrows
$\leftrightarrows$: \leftrightarrows	$\rightleftarrows$: \rightleftarrows
$\Lleftarrow$: \Lleftarrow	$\Rrightarrow$: \Rrightarrow
$\twoheadleftarrow$: \twoheadleftarrow	$\twoheadrightarrow$: \twoheadrightarrow
$\leftarrowtail$: \leftarrowtail	$\rightarrowtail$: \rightarrowtail
$\leftrightharpoons$: \leftrightharpoons	$\rightleftharpoons$: \rightleftharpoons
$\Lsh$: \Lsh	$\Rsh$: \Rsh
$\looparrowleft$: \looparrowleft	$\looparrowright$: \looparrowright
$\curvearrowleft$: \curvearrowleft	$\curvearrowright$: \curvearrowright
$\circlearrowleft$: \circlearrowleft	$\circlearrowright$: \circlearrowright
$\multimap$: \multimap	$\upuparrows$: \upuparrows
$\downdownarrows$: \downdownarrows	$\upharpoonleft$: \upharpoonleft
$\upharpoonright$: \upharpoonright	$\downharpoonright$: \downharpoonright
$\rightsquigarrow$: \rightsquigarrow	$\leftrightsquigarrow$: \leftrightsquigarrow

7.18 AMS否定式二元关系和箭头

$\nless$: \nless	$\ngtr$: \ngtr	$\varsubsetneqq$: \varsubsetneqq
$\lneq$: \lneq	$\gneq$: \gneq	$\varsupsetneqq$: \varsupsetneqq
$\nleq$: \nleq	$\ngeq$: \ngeq	$\nsubseteqq$: \nsubseteqq
$\nleqslant$: \nleqslant	$\ngeqslant$: \ngeqslant	$\nsupseteqq$: \nsupseteqq
$\lneqq$: \lneqq	$\gneqq$: \gneqq	$\nmid$: \nmid
$\lvertneqq$: \lvertneqq	$\gvertneqq$: \gvertneqq	$\nparallel$: \nparallel
$\nleqq$: \nleqq	$\ngeqq$: \ngeqq	$\nshortmid$: \nshortmid
$\lnsim$: \lnsim	$\gnsim$: \gnsim	$\nshortparallel$: \nshortparallel
$\lnapprox$: \lnapprox	$\gnapprox$: \gnapprox	$\nsim$: \nsim
$\nprec$: \nprec	$\nsucc$: \nsucc	$\ncong$: \ncong
$\npreceq$: \npreceq	$\nsucceq$: \nsucceq	$\nvdash$: \nvdash
$\precneqq$: \precneqq	$\succneqq$: \succneqq	$\nvDash$: \nvDash
$\precnsim$: \precnsim	$\succnsim$: \succnsim	$\nVdash$: \nVdash
$\precnapprox$: \precnapprox	$\succnapprox$: \succnapprox	$\nVDash$: \nVDash
$\subsetneq$: \subsetneq	$\supsetneq$: \supsetneq	$\ntriangleleft$: \ntriangleleft
$\varsubsetneq$: \varsubsetneq	$\varsupsetneq$: \varsupsetneq	$\ntriangleright$: \ntriangleright
$\nsubseteq$: \nsubseteq	$\nsupseteq$: \nsupseteq	$\ntrianglelefteq$: \ntrianglelefteq
$\subsetneqq$: \subsetneqq	$\supsetneqq$: \supsetneqq	$\ntrianglerighteq$: \ntrianglerighteq
$\nleftarrow$: \nleftarrow	$\nrightarrow$: \nrightarrow	$\nleftrightarrow$: \nleftrightarrow
$\nLeftarrow$: \nLeftarrow	$\nRightarrow$: \nRightarrow	$\nLeftrightarrow$: \nLeftrightarrow

7.19 AMS各式各样符号

$\hbar$: \hbar	$\hslash$: \hslash	$\Bbbk$: \Bbbk
$\square$: \square	$\blacksquare$: \blacksquare	$\circledS$: \circledS
$\vartriangle$: \vartriangle	$\blacktriangle$: \blacktriangle	$\complement$: \complement
$\triangledown$: \triangledown	$\blacktriangledown$: \blacktriangledown	$\Game$: \Game
$\lozenge$: \lozenge	$\blacklozenge$: \blacklozenge	$\bigstar$: \bigstar
$\angle$: \angle	$\measuredangle$: \measuredangle	
$\diagup$: \diagup	$\diagdown$: \diagdown	$\backprime$: \backprime
$\nexists$: \nexists	$\Finv$: \Finv	$\varnothing$: \varnothing
$\eth$: \eth	$\sphericalangle$: \sphericalangle	$\mho$: \mho