Markdown Basics

Overview

Quarto is based on Pandoc and uses its variation of markdown as its underlying document syntax. Pandoc markdown is an extended and slightly revised version of John Gruber's Markdown syntax.

Markdown is a plain text format that is designed to be easy to write, and, even more importantly, easy to read:

> A Markdown-formatted document should be publishable as-is, as plain text, without looking like it's been marked up with tags or formatting instructions. – John Gruber

This document provides examples of the most commonly used markdown syntax. See the full documentation of Pandoc's Markdown for more in-depth documentation.

Text Formatting

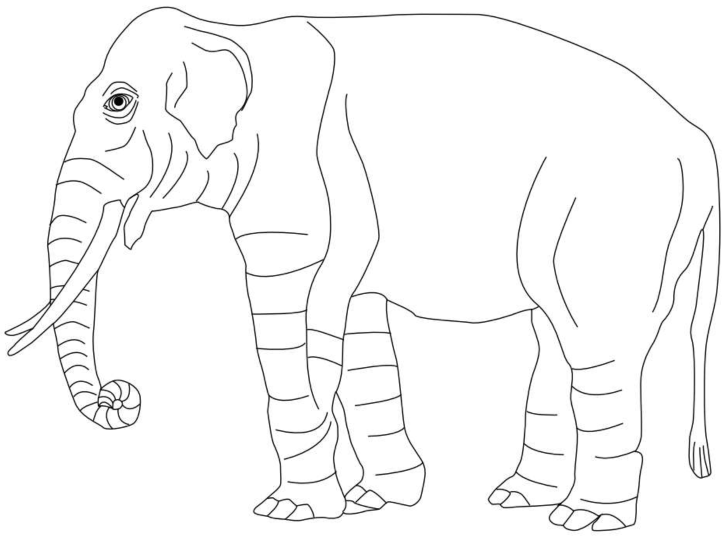
Markdown Syntax	Output
`*italics*, **bold**, ***bold italics***`	*italics*, **bold**, ***bold italics***
`superscript^2^ / subscript~2~`	superscript^2 / subscript$_2$
`~~strikethrough~~`	~~strikethrough~~
`` `verbatim code` ``	`verbatim code`

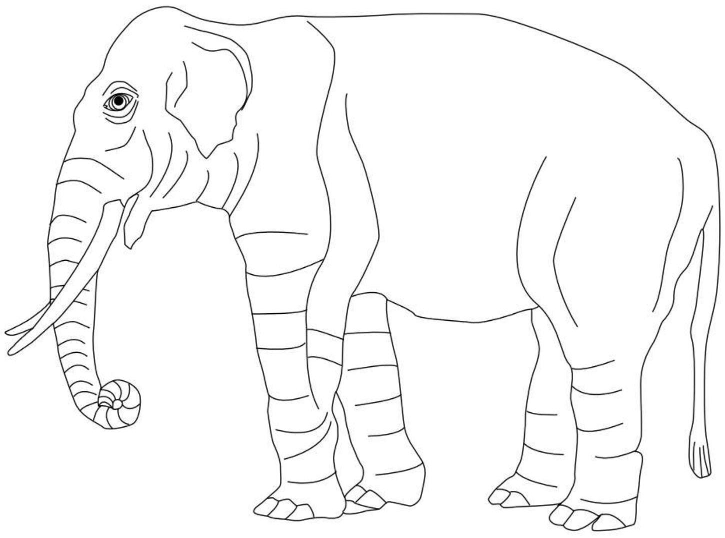
Headings

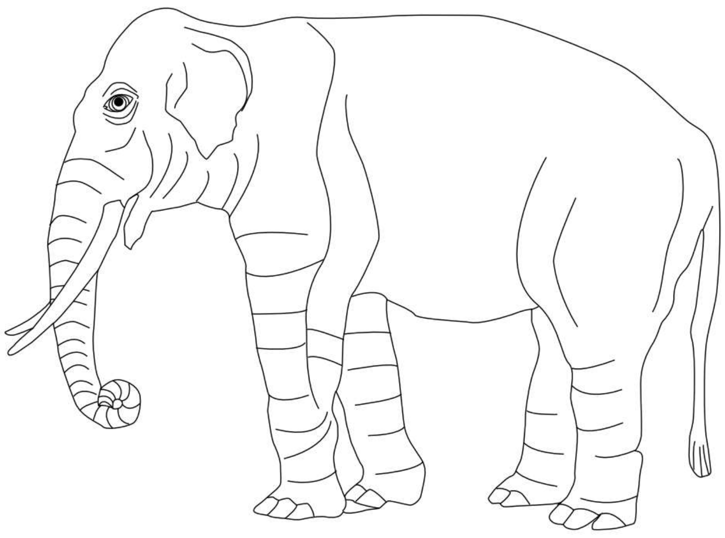
Markdown Syntax	Output
`# Header 1`	**Header 1**
`## Header 2`	**Header 2**
`### Header 3`	**Header 3**
`#### Header 4`	Header 4
`##### Header 5`	Header 5
`###### Header 6`	Header 6

Links & Images

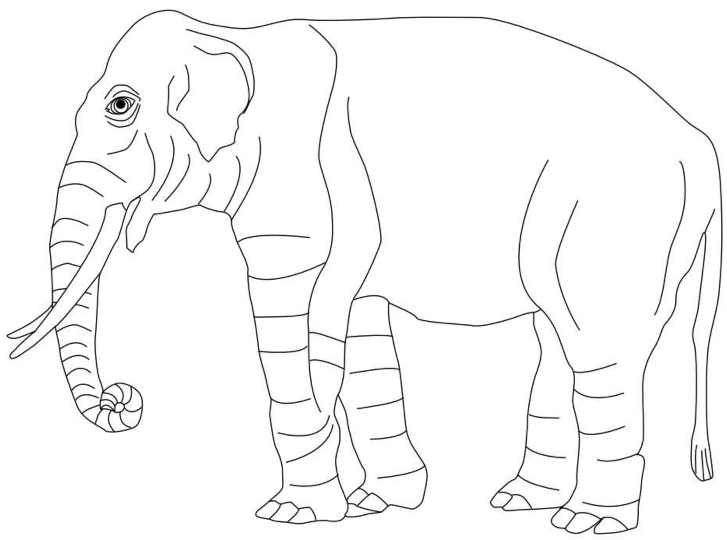
Markdown Syntax	Output
`<https://quarto.org>`	https://quarto.org
`[Quarto](https://quarto.org)`	Quarto

Markdown Syntax	Output
`![Caption](elephant.png)`	 Caption

`[![Caption](elephant.png)]`
`(https://quarto.org)`
	

`[![Caption](elephant.png "An elephant"`
`(https://quarto.org)`
	

Markdown Syntax	Output

`[![](elephant.png){fig-alt="Alt text"}`
`(https://quarto.org)`
	

Lists

Markdown Syntax	Output

`* unordered list`
`    + sub-item 1`
`    + sub-item 2`
`        - sub-sub-item 1`
	• unordered list
	∘ sub-item 1
	∘ sub-item 2
	▪ sub-sub-item 1

`*   item 2`
 	
`    Continued (indent 4 spaces)`
	• item 2
	
	Continued (indent 4 spaces)

`1. ordered list`
`2. item 2`
`    i) sub-item 1`
`         A.  sub-sub-item 1`
	1. ordered list
	2. item 2
	i. sub-item 1
	A. sub-sub-item 1

`- [ ] Task 1`
`- [x] Task 2`
	☐ Task 1
	☑ Task 2

Markdown Syntax	Output
`(@)  A list whose numbering` `continues after` `(@)  an interruption`	1. A list whose numbering continues after 2. an interruption
`::: {}` `1. A list` `:::` `::: {}` `1. Followed by another list` `:::`	1. A list 1. Followed by another list
`term` `: definition`	**term** definition

Note that unlike other Markdown renderers (notably Jupyter and GitHub), lists in Quarto require an entire blank line above the list. Otherwise the list will not be rendered in list form, rather it will all appear as normal text along a single line.

Tables

Markdown Syntax

```
| Right | Left | Default | Center |
|------:|:-----|---------|:------:|
|   12  |  12  |    12   |    12  |
|  123  |  123 |   123   |   123  |
|    1  |    1 |     1   |     1  |
```

Output

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

Right	Left	Default	Center
1	1	1	1

Learn more in the article on Tables.

Source Code

Use `` ``` `` to delimit blocks of source code:

~~~
```
code
```
~~~

Add a language to syntax highlight code blocks:

~~~
```python
1 + 1
```
~~~

Pandoc supports syntax highlighting for over 140 different languages. If your language is not supported then you can use the `default` language to get a similar visual treatment:

~~~
```default
code
```
~~~

Equivalent to the short form used in the examples above is a longer form that uses the language as a class (i.e. `.python`) inside braces:

~~~
```{.python}
1 + 1
```
~~~

The longer form allows you to add attributes to the block in a similar way to Divs. Some specific features that use this syntax are Lines Numbers and Code Filename. Here is an example of the latter:

~~~
```{.python filename="run.py"}
code
```
~~~

If you are creating HTML output there is a wide variety of options available for code block output. See the article on HTML Code for additional details.

Raw Content

Raw content can be included directly without Quarto parsing it using Pandoc's raw attribute. A raw block starts with ` ```{= ` followed by a format and closing `}`, e.g. here's a raw HTML block:

~~~
```{=html}
<iframe src="https://quarto.org/" width="500" height="400"></iframe>
```
~~~

For PDF output use a raw LaTeX block:

~~~
```{=latex}
\renewcommand*{\labelitemi}{\textgreater}
```
~~~

As another example, if you are using the Typst format you can use a raw block to include Typst syntax:

~~~
```{=typst}
#set text(fill: red)
This text is red.
```
~~~

You can also include raw content inline:

```
Here's some raw inline HTML: `<a>html</a>`{=html}
```

Equations

Use `$` delimiters for inline math and `$$` delimiters for display math. For example:

Markdown Syntax	Output
`inline math: $E = mc^{2}$`	inline math: $E = mc^{2}$

`display math:`
 	
 `$$E = mc^{2}$$`
	display math: $$E = mc^{2}$$

If you want to define custom TeX macros, include them within `$$` delimiters enclosed in a `.hidden` block. For example:

```
::: {.hidden}
$$
 \def\RR{{\bf R}}
 \def\bold#1{{\bf #1}}
$$
:::
```

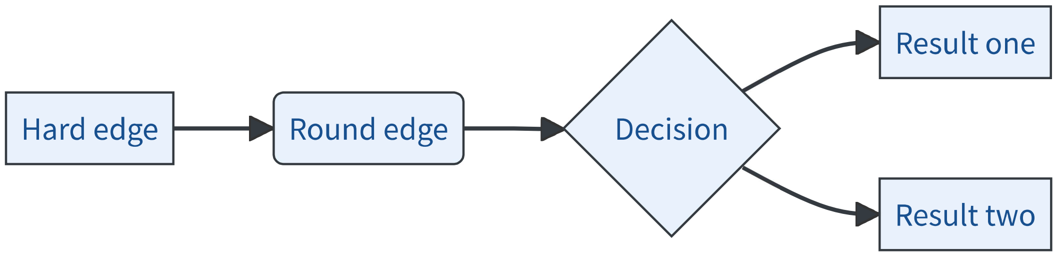
For HTML math processed using MathJax (the default) you can use the `\def`, `\newcommand`, `\renewcommand`, `\newenvironment`, `\renewenvironment`, and `\let` commands to create your own macros and environments.

Diagrams

Quarto has native support for embedding Mermaid and Graphviz diagrams. This enables you to create flowcharts, sequence diagrams, state diagrams, Gantt charts, and more using a plain text syntax inspired by markdown.

For example, here we embed a flowchart created using Mermaid:

~~~
```{mermaid}
flowchart LR
  A[Hard edge] --> B(Round edge)
  B --> C{Decision}
  C --> D[Result one]
  C --> E[Result two]
```
~~~



Learn more in the article on Diagrams.

Videos

You can include videos in documents using the `{{< video >}}` shortcode. For example, here we embed a YouTube video:

```
{{< video https://www.youtube.com/embed/wo9vZccmqwc >}}
```

Videos can refer to video files (e.g. MPEG) or can be links to videos published on YouTube, Vimeo, or Brightcove. Learn more in the article on Videos.

Page Breaks

The `pagebreak` shortcode enables you to insert a native pagebreak into a document (.e.g in LaTeX this would be a `\newpage`, in MS Word a docx-native pagebreak, in HTML a `page-break-after: always` CSS directive, etc.):

```
page 1

{{< pagebreak >}}

page 2
```

Native pagebreaks are supported for HTML, LaTeX, Context, MS Word, Open Document, and ePub (for other formats a form-feed character `\f` is inserted).

Divs and Spans

You can add classes, attributes, and other identifiers to regions of content using Divs and Spans (you'll see an example of this below in Callout Blocks).

For example, here we add the "border" class to a region of content using a div (`:::`):

```
::: {.border}
This content can be styled with a border
:::
```

Once rendered to HTML, Quarto will translate the markdown into:

```
<div class="border">
  <p>This content can be styled with a border</p>
</div>
```

Divs start with a fence containing at least three consecutive colons plus some attributes. The attributes may optionally be followed by another string of consecutive colons. The Div ends with another line containing a string of at least three consecutive colons. The Div should be separated by blank lines from preceding and following blocks. Divs may also be nested. For example

```
::::: {#special .sidebar}

::: {.warning}
Here is a warning.
:::

More content.
:::::
```

Once rendered to HTML, Quarto will translate the markdown into:

```html
<div id="special" class="sidebar">
  <div class="warning">
    <p>Here is a warning.</p>
  </div>
  <p>More content.</p>
</div>
```

Fences without attributes are always closing fences. Unlike with fenced code blocks, the number of colons in the closing fence need not match the number in the opening fence. However, it can be helpful for visual clarity to use fences of different lengths to distinguish nested divs from their parents.

A bracketed sequence of inlines, as one would use to begin a link, will be treated as a `Span` with attributes if it is followed immediately by attributes:

```
[This is *some text*]{.class key="val"}
```

Once rendered to HTML, Quarto will translate the markdown into:

```html
<span class="class" data-key="val">
  This is <em>some text</em>
</span>
```

Typically, you'll use CSS and/or a Filter along with Divs and Spans to provide styling or other behavior within rendered documents.

Ordering of Attributes

Both divs and spans in Pandoc can have any combination of identifiers, classes, and (potentially many) key-value attributes. In order for these to be recognized by Pandoc, they have to be provided in a specific order: identifiers, classes, and then key-value attributes. Any of these can be omitted, but must follow that order if they are provided. For example, the following is valid:

```
[This is good]{#id .class key1="val1" key2="val2"}
```

However, the following *will not be recognized by Pandoc*:

```
[This does *not* work!]{.class key="val" #id}
```

This ordering restriction applies to both divs and spans. See Pandoc's documentation on Divs and Spans for additional details.

Callout Blocks

Markdown Syntax

```
:::{.callout-note}
Note that there are five types of callouts, including:
`note`, `tip`, `warning`, `caution`, and `important`.
:::
```

Output

Note

Note that there are five types of callouts, including `note`, `tip`, `warning`, `caution`, and `important`.

Learn more in the article on Callout Blocks.

Other Blocks

Markdown Syntax	Output
`> Blockquote`	> Blockquote

`::: {.classname}`
`Div`
`:::`
	Div

`| Line Block`
`|    Spaces and newlines`
`|    are preserved`
	Line Block
	  Spaces and newlines
	  are preserved

Special Characters

Markdown Syntax	Output
`endash: --`	endash: –
`emdash: ---`	emdash: —

Keyboard Shortcuts

The `kbd` shortcode can be used to describe keyboard shortcuts in documentation. On Javascript formats, it will attempt to detect the operating system of the format and show the correct shortcut. On print formats, it will print the keyboard shortcut information for all operating systems.

For example, writing the following markdown:

```
To print, press {{< kbd Shift-Ctrl-P >}}. To open an existing new project, press {{< kbd
         mac=Shift-Command-O win=Shift-Control-O linux=Shift-Ctrl-L >}}.
```

will render the keyboard shortcuts as:

To print, press `Shift-Ctrl-P`. To open an existing new project, press `Shift-Control-O`.