Quarto Demonstration

Overview

Quarto is based on Pandoc and uses its variation of markdown as its underlying document syntax. 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

The usual formatting features are supported, such as *italics*, **bold**, ***bold italics***, superscript^2, subscript$_2$, ~~striketrough~~, and `verbatim code`.

Headings

Headings are defined using an increansingly number of `#`, going deeper in level.

Links and images

`<https://quarto.org>` translates to https://quarto.org. To define a custom label, use the syntax `[Quarto](http://quarto.org)`, which outputs to Quarto.

To embed a image, use link prefixed with `!`, such as `![Elephant](elephant.png)`.

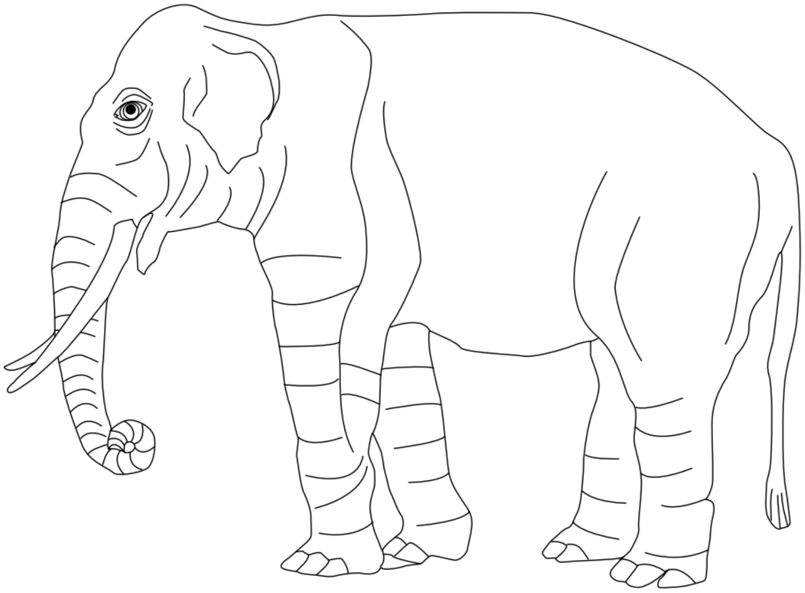


Figure 1: Elephant

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

`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

`1. ordered list`
`2. item 2`
 	
``
```
python ``
`print("Hello, World!")`
``
```
 ``
`A.  sub-sub-item 1`
	1. ordered list
	2. item 2
	`print("Hello, World!")`
	A. sub-sub-item 1

Markdown Syntax	Output

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

`term`
`: definition`
	term definition

Footnotes

There are many ways to number and format footnotes supported by Pandoc:

```
Here is a footnote,[^1] and here's another.[^longnote]

[^1]: A simple and concise footnote.

[^longnote]: And here's a footnote with multiple blocks.

    Subsequent paragraphs can be indented so that they are included
in the previous footnote.

        { some.code }

    The whole paragraph can be indented, or just the first line, in
    a similar way to multi-paragraph list items.

This paragraph has no indentation, and so is not part of the note.
```

The above example generates the following output:

Here is a footnote,[1] and here's another.[2]

This paragraph has no indentation, and so is not part of the note.

Footnotes may also be written inline:

[1] A simple and concise footnote.

[2] And here's a footnote with multiple blocks.

Subsequent paragraphs can be indented so that they are included in the previous footnote.

```
{ some.code }
```

The whole paragraph can be indented, or just the first line, in a similar way to multi-paragraph list items.

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

Which generates the following output:

Here is an inline note.[3]

> ⚠ Footnote IDs must be unique
>
> Footnote identifiers, e.g., the 1 in ^1, need to be unique within a document. In Quarto books, chapters are combined into a single document for certain formats (including PDF, DOCX, and EPUB), so footnote identifiers need to be unique across chapters.

Tables

Simple tables can be written with the following syntax:

```
| Right | Left | Default | Center |
|------:|:-----|---------|:------:|
|  12   |  12  |    12   |   12   |
| 123   |  123 |  1233   |   123  |
|   1   |   1  |     1   | 1      |
```

With the result:

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

Source Code

Use ``` to delimit blocks of source code. A language can be specified for syntax highlight, or the `default` can be used if your language is not supported.

[3] Inlines notes are easier to write, since you don't have to pick an identifier and move down to type the note.

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

The language can also be specified as a class attribute, like `{.python}`. This form allows some specific features such as lines numbers and code filename.

~~~
```{.python code-line-numbers="true" filename="run.py"}
def main():
  print("Hello world!")
  simple_sum = 2 + 2
  print(simple_sum)
```
~~~

Listing 1 `run.py`

```python
def main():
  print("Hello world!")
  simple_sum = 2 + 2
  print(simple_sum)
```

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}
\LaTeX
```
~~~

LaTeX

Output of Raw Content

Mind that raw content is only visible in the relevant output. In these examples, HTML raw content is not visible on PDF output, and vice versa.

Equations

Use `$` to delimit inline math, and `$$` for display math.

Markdown Syntax	Output

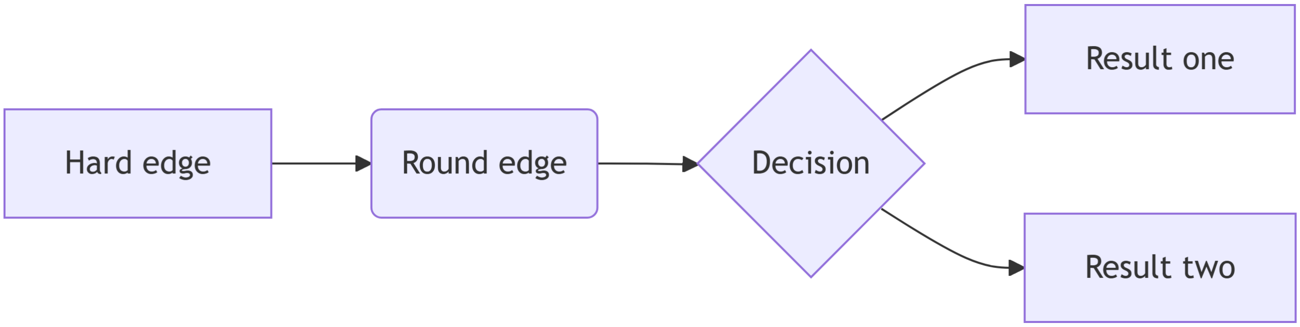
`inline math: $E = mc^{2}$`
`display math:`
`$$E = mc^{2}$$`
	inline math: $E = mc^2$
	display math:
	$E = mc^2$

Custom TeX macros can be defined inside `$$` delimiters enclosed in a `.hidden` block. 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 Mermaid and Graphviz diagrams. For example:

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



Videos

Videos can be included using the `{{< video >}}` shortcode. For example, embeding an YouTube video:

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

https://www.youtube.com/embed/wo9vZccmqwc

Videos can be local files, or links to YouTube, Vimeo or Brightcove.

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

Divs and Spans can be used to add attributes, classes and other identifers to arbitrary regions of content. Despite the nomenclature coming from HTML, the syntax is used in Quarto across output formats. For example, Callout Blocks are specified using div syntax, and Small Caps are specified using span syntax.

Divs

To add a border to a region we can use the `.border` class:

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

This content can be styled with a border

Spans

A bracketed sequence of inlines will be treated as a Span if followed immediately by attributes:

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

When rendered to HTML will generate the output:

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

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.

Callout Blocks

Markdown Syntax

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

Output

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

Other Blocks

Markdown Syntax	Output
`> Blockquote`	Blockquote

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

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

Other Spans

To create text in small caps, that is underlined, or that is highlighted, use a span with one of the classes `.smallcaps`, `.underline` or `.mark` respectively.

Markdown Syntax	Output
`[This text is smallcaps]{.smallcaps}`	THIS TEXT IS SMALLCAPS
`[This text is underlined]{.underline}`	This text is underlined
`[This text is highlighted]{.mark}`	This text is highlighted

> ℹ In supported formats only
>
> Support for these classes comes directly from Pandoc. Not all formats support all of these classes. In particular, `.mark` is not currently supported in `format: pptx`.

Special Characters

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

Keyboard Shortcuts

The `kbd` shortcode can be used to describe keyboard shortcuts in documentation:

- single keyboard shortcuts for every operating system use a **positional parameter**: `{{< kbd Ctrl-C >}}`
- different shortcuts for different operating systems use **keyword parameters `win`, `mac`, and `linux`**: `{{< kbd mac=Shift-Command-O win=Shift-Control-O linux=Shift-Ctrl-L >}}`

On Javascript formats, Quarto 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` (windows), `Shift-Command-O` (mac), `Shift-Ctrl-L` (linux).