localhost

Writing technical content in Academic

Maghimai Marcus

6-7 minutes

Academic is designed to give technical content creators a seamless experience. You can focus on the content and Academic handles the rest.

Highlight your code snippets, take notes on math classes, and draw diagrams from textual representation.

On this page, you'll find some examples of the types of technical content that can be rendered with Academic.

Examples

Code

Academic supports a Markdown extension for highlighting code syntax. You can enable this feature by toggling the `highlight` option in your `config/_default/params.toml` file.

```python
import pandas as pd
data = pd.read_csv("data.csv")
data.head()
```

renders as

```
import pandas as pd
data = pd.read_csv("data.csv")
data.head()
```

Charts

Academic supports the popular Plotly chart format.

Save your Plotly JSON in your page folder, for example `chart.json`, and then add the `{{< chart data="chart" >}}` shortcode where you would like the chart to appear.

Demo:

You might also find the Plotly JSON Editor useful.

Math

Academic supports a Markdown extension for math. You can enable this feature by toggling the `math` option in your `config/_default/params.toml` file.

To render *inline* or *block* math, wrap your LaTeX math with `$...$` or `$$...$$`, respectively.

Example **math block**:

```
$$\gamma_{n} = \frac{
\left | \left (\mathbf x_{n} - \mathbf x_{n-1}
\right )^T
\left [\nabla F (\mathbf x_{n}) - \nabla F
(\mathbf x_{n-1}) \right ] \right |}
{\left \|\nabla F(\mathbf{x}_{n}) - \nabla
F(\mathbf{x}_{n-1}) \right \|^2}$$
```

renders as

Example **inline math** `$\nabla F(\mathbf{x}_{n})$` renders as .

Example **multi-line math** using the `\\\\` math linebreak:

```
$$f(k;p_{0}^{*}) = \begin{cases}p_{0}^{*} &
\text{if }k=1, \\\\
1-p_{0}^{*} & \text{if }k=0.\end{cases}$$
```

renders as

Diagrams

Academic supports a Markdown extension for diagrams. You can enable this feature by toggling the `diagram` option in your `config/_default/params.toml` file or by adding `diagram: true` to your page front matter.

An example **flowchart**:

````
```mermaid
graph TD
A[Hard] -->|Text| B(Round)
B --> C{Decision}
C -->|One| D[Result 1]
C -->|Two| E[Result 2]
```
````

renders as

graph TD A[Hard] -->|Text| B(Round) B --> C{Decision} C -->|One| D[Result 1] C -->|Two| E[Result 2]

An example **sequence diagram**:

```mermaid
sequenceDiagram
Alice->>John: Hello John, how are you?
loop Healthcheck
    John->>John: Fight against hypochondria
end
Note right of John: Rational thoughts!
John-->>Alice: Great!
John->>Bob: How about you?
Bob-->>John: Jolly good!
```

renders as

sequenceDiagram Alice->>John: Hello John, how are you? loop Healthcheck John->>John: Fight against hypochondria end Note right of John: Rational thoughts! John-->>Alice: Great! John->>Bob: How about you? Bob-->>John: Jolly good!

An example **Gantt diagram**:

```mermaid
gantt
section Section
Completed :done,    des1, 2014-01-06,2014-01-08
Active        :active,  des2, 2014-01-07, 3d
Parallel 1   :         des3, after des1, 1d
Parallel 2   :         des4, after des1, 1d
Parallel 3   :         des5, after des3, 1d
Parallel 4   :         des6, after des4, 1d
```

renders as

gantt section Section Completed :done, des1, 2014-01-06,2014-01-08 Active :active, des2, 2014-01-07, 3d Parallel 1 : des3, after des1, 1d Parallel 2 : des4, after des1, 1d Parallel 3 : des5, after des3, 1d Parallel 4 : des6, after des4, 1d

An example **class diagram**:

```mermaid
classDiagram
Class01 <|-- AveryLongClass : Cool
<<interface>> Class01
Class09 --> C2 : Where am i?
Class09 --* C3
Class09 --|> Class07
Class07 : equals()
Class07 : Object[] elementData
Class01 : size()
Class01 : int chimp
Class01 : int gorilla
class Class10 {
  <<service>>
  int id
  size()
}
```

renders as

classDiagram Class01 <|-- AveryLongClass : Cool <<interface>> Class01 Class09 --> C2 : Where am i? Class09 --* C3 Class09 --|> Class07 Class07 : equals() Class07 : Object[] elementData Class01 : size() Class01 : int chimp Class01 : int gorilla class

Class10 { <<service>> int id size() }

An example **state diagram**:

```mermaid
stateDiagram
[*]
Still
Still
Moving
Moving
Crash
```

renders as

stateDiagram [*] --> Still Still --> [*] Still --> Moving Moving --> Still Moving --> Crash Crash --> [*]

Todo lists

You can even write your todo lists in Academic too:

```
- [x] Write math example
- [x] Write diagram example
- [ ] Do something else
```

renders as

- Write math example
- Write diagram example
- Do something else

Tables

Represent your data in tables:

```
| First Header  | Second Header |
| ------------- | ------------- |
| Content Cell  | Content Cell  |
| Content Cell  | Content Cell  |
```

renders as

First Header	Second Header
Content Cell	Content Cell
Content Cell	Content Cell

Callouts

Academic supports a shortcode for callouts, also referred to as *asides*, *hints*, or *alerts*. By wrapping a paragraph in `{{% callout note %}} ... {{% /callout %}}`, it will render as an aside.

```
{{% callout note %}}
A Markdown aside is useful for displaying
notices, hints, or definitions to your readers.
{{% /callout %}}
```

renders as

A Markdown aside is useful for displaying notices, hints, or definitions to your readers.

Spoilers

Add a spoiler to a page to reveal text, such as an answer to a question, after a button is clicked.

```
{{< spoiler text="Click to view the spoiler" >}}
```

```
You found me!
{{< /spoiler >}}
```

renders as

▶ Click to view the spoiler

Icons

Academic enables you to use a wide range of icons from *Font Awesome* and *Academicons* in addition to emojis.

Here are some examples using the `icon` shortcode to render icons:

```
{{< icon name="terminal" pack="fas" >}} Terminal
{{< icon name="python" pack="fab" >}} Python
{{< icon name="r-project" pack="fab" >}} R
```

renders as

Terminal
Python
R

Did you find this page helpful? Consider sharing it 🙌