1 Markdown to LaTeX Comprehensive Guide

This guide demonstrates all supported markdown syntax features for the md-to-luatex converter.

1.1 Headings

```markdown
% % % %
# Heading 1
## Heading 2
### Heading 3
#### Heading 4
##### Heading 5
###### Heading 6
```

2 Heading 1

2.1 Heading 2

2.1.1 Heading 3

Heading 4

Heading 5

Heading 6

2.2 Text Formatting

2.2.1 Bold

```markdown
**bold text** or __bold text__
```

bold text or **bold text**

2.2.2 Italic

```markdown
_italic text_ or *italic text*
```

italic text or *italic text*

2.2.3 Bold + Italic

```markdown
**_bold and italic_** or ***bold and italic***
```

bold and italic or ***bold and italic***

2.2.4 Strikethrough

```markdown
~~strikethrough text~~
```

~~strikethrough text~~

2.2.5 Highlight

```markdown
==highlighted text==
```

highlighted text

2.2.6 Superscript

```markdown
^superscript^ (e.g., x^2^)
```

superscript (e.g., x^{2})

2.2.7 Subscript

```markdown
~subscript~ (e.g., H~2~O)
```

subscript (e.g., $H_{2}O$)

2.3 Inline Code

```markdown
`inline code`
```

`inline code`

2.4 Code Blocks

2.4.1 Fenced Code Block

```markdown
```python
def hello():
    print("Hello, World!")
```
```

```python
def hello():
    print("Hello, World!")
```

2.4.2 Code Block Without Language

markdown

```
```
plain text code block
```
```

```
plain text code block
```

2.4.3 Terminal Block

markdown

```
```terminal
$ command
output result
```
```

text

```
$ command
output result
```

2.5 Links

markdown

```
[link text](https://example.com)
```

link text

markdown

```
[link text](https://example.com "title")
```

link text

2.6 Images

markdown

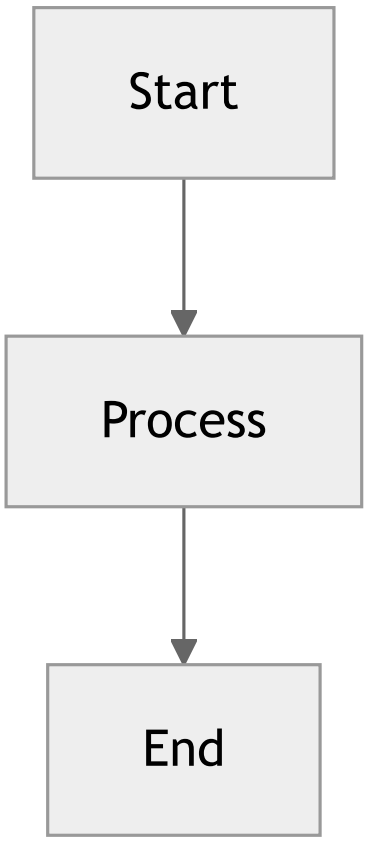
```
![alt text](image.png)
```

```markdown
![alt text](image.png "title")
```

Figure 1: title

2.7 Mermaid Diagrams

```markdown
```mermaid
graph TD
    A[Start] --> B[Process]
    B --> C[End]
```
```



2.8 Lists

2.8.1 Unordered List

```markdown
- Item 1
- Item 2
  - Nested item 2.1
  - Nested item 2.2
- Item 3
```

- Item 1
- Item 2
  - Nested item 2.1
  - Nested item 2.2
- Item 3

2.8.2 Ordered List

```markdown
1. First item
2. Second item
   1. Nested item 2.1
   2. Nested item 2.2
3. Third item
```

1. First item
2. Second item
    1. Nested item 2.1
    2. Nested item 2.2
3. Third item

2.8.3 Task Lists

```markdown
- [x] Completed task
- [ ] Incomplete task
- [/] Partially completed task
```

- [x] Completed task
- [ ] Incomplete task
- [/] Partially completed task

2.9 Definition Lists

```markdown
Term 1
: Definition 1

Term 2
: Definition 2a
: Definition 2b
```

Term 1
: Definition 1

Term 2
: Definition 2a
: Definition 2b

2.10 Blockquotes

```markdown
> *This is a blockquote.*
> *It can span multiple lines.*
>
> *And multiple paragraphs.*
```

> This is a blockquote. It can span multiple lines.
> And multiple paragraphs.

2.10.1 Nested Blockquotes

```markdown
> *Level 1*
>
> > *Level 2*
> >
> > > *Level 3*
```

> Level 1
>
> > Level 2
> >
> > > Level 3

2.10.2 Blockquotes with Styled Content

Blockquotes can contain inline code, keyboard shortcuts, tables, and other formatted elements:

```markdown
> This blockquote contains `inline code` and keyboard shortcuts like [[Ctrl] +
↪  [C]].
>
> It can also have **bold**, _italic_, and ==highlighted== text.
>
> | Feature | Supported |
> | ------- | --------- |
> | Tables  | Yes       |
> | Code    | Yes       |
```

> This blockquote contains `inline code` and keyboard shortcuts like Ctrl + C.
> It can also have **bold**, *italic*, and highlighted text.
>
> | Feature | Supported |
> | --- | --- |
> | Tables | Yes |
> | Code | Yes |

2.10.3 Blockquotes with Code Blocks

```markdown
> Here's a code example inside a blockquote:
>
> ```python
> def greet(name):
>     return f"Hello, {name}!"
> ```
>
> The code block maintains its syntax highlighting.
```

> Here's a code example inside a blockquote:
>
> ```python
> def greet(name):
>     return f"Hello, {name}!"
> ```
>
> The code block maintains its syntax highlighting.

2.11 GitHub Alerts

2.11.1 Note Alert

```markdown
> [!NOTE]
> This is a note alert with blue styling.
```

> **NOTE**
> This is a note alert with blue styling.

2.11.2 Tip Alert

```markdown
> [!TIP]
> This is a tip alert with green styling.
```

> **TIP**
> This is a tip alert with green styling.

2.11.3 Important Alert

```markdown
> [!IMPORTANT]
> This is an important alert with purple styling.
```

> **IMPORTANT**
> This is an important alert with purple styling.

2.11.4 Warning Alert

```markdown
> [!WARNING]
> This is a warning alert with yellow/orange styling.
```

> **WARNING**
> This is a warning alert with yellow/orange styling.

2.11.5 Caution Alert

```markdown
> [!CAUTION]
> This is a caution alert with red styling.
```

> **CAUTION**
> This is a caution alert with red styling.

2.11.6 Alerts with Rich Content

GitHub alerts can also contain formatted text, code, tables, and keyboard shortcuts:

```markdown
> [!TIP]
> **Pro Tip:** Use `git commit -m "message"` to commit changes.
>
> Common keyboard shortcuts:
>
> - Save: [[Ctrl] + [S]]
> - Undo: [[Ctrl] + [Z]]
>
> | Command      | Description             |
> | ------------ | ----------------------- |
> | `git status` | Check repository status |
> | `git log`    | View commit history     |
```

> **TIP**
> **Pro Tip:** Use `git commit -m "message"` to commit changes.
> Common keyboard shortcuts:
>
> - Save: Ctrl + S
> - Undo: Ctrl + Z
>
> | Command | Description |
> |---|---|
> | `git status` | Check repository status |
> | `git log` | View commit history |

```markdown
> [!WARNING]
> This contains ==highlighted text==, ~~strikethrough~~, and ^superscript^!
>
> Code example:
>
> ```bash
> rm -rf /
> ```
>
> **Never** run the above command!
```

> **WARNING**
> This contains highlighted text, ~~strikethrough~~, and superscript!
> Code example:
>
> ```bash
> rm -rf /
> ```
>
> **Never** run the above command!

2.12 Tables

2.12.1 Pipe Tables

```markdown
| Header 1 | Header 2 | Header 3 |
| -------- | -------- | -------- |
| Cell 1   | Cell 2   | Cell 3   |
| Cell 4   | Cell 5   | Cell 6   |
```

Header 1	Header 2	Header 3
Cell 1	Cell 2	Cell 3
Cell 4	Cell 5	Cell 6

2.12.2 Table Alignment

```markdown
| Left | Center | Right |
| :--- | :----: | ----: |
| L1   |   C1   |    R1 |
| L2   |   C2   |    R2 |
```

Left	Center	Right
L1	C1	R1
L2	C2	R2

2.13 Horizontal Rule

```markdown
---
```

```markdown
---
```

```markdown
***
```

2.14 Math Expressions

2.14.1 Inline Math

```markdown
This is inline math: $E = mc^2$
```

This is inline math: $E = mc^2$

2.14.2 Display Math

```markdown
$$
\int_{0}^{\infty} e^{-x^2} dx = \frac{\sqrt{\pi}}{2}
$$
```

$$\int_{0}^{\infty} e^{-x^2} dx = \frac{\sqrt{\pi}}{2}$$

2.15 Footnotes

2.15.1 Inline Footnote

```markdown
Text with inline footnote^[This is the footnote content].
```

Text with inline footnote[1] .

2.15.2 Reference Footnote

```markdown
Text with reference footnote[^1].

[^1]: This is the footnote content.
```

Text with reference footnote[2] .

2.16 Keyboard Shortcuts

2.16.1 Single Key

```markdown
[[Ctrl]]
```

Ctrl

2.16.2 Key Combination with Plus

```markdown
[[Ctrl] + [C]]
```

Ctrl + C

2.16.3 Key Combination with Minus

```markdown
[[Alt] - [Tab]]
```

Alt - Tab

[1] This is the footnote content

[2] This is the footnote content.

2.17 HTML Support

2.17.1 Line Break

```markdown
Line 1<br>
Line 2
```

Line 1
Line 2

```markdown
Line 1<br/>
Line 2
```

Line 1
Line 2

2.18 Executable Python Code Blocks

The converter supports executing Python code blocks directly within your markdown and including their output or generated plots in the final PDF.

2.18.1 Prerequisites

For Python code execution to work, you need Python installed on your system. To generate plots with matplotlib, install the required packages:

```powershell
python -m pip install matplotlib numpy
```

> **NOTE**
> Mermaid diagram support is already documented in the Mermaid Diagrams section and requires separate installation of `@mermaid-js/mermaid-cli` via npm.

2.18.2 Basic Syntax

Mark a Python code block for execution using properties in curly braces:

````markdown
```python {.execute}
print("Hello, World!")
```
````

2.18.3 Available Properties

- `.execute` - Execute the code block (required)
- `.show-code` - Display the source code in the output
- `.show-output` - Display execution output/plot (enabled by default)
- `.hide-code` - Explicitly hide the source code (default)
- `.hide-output` - Hide execution output/plot

2.18.4 Property Combinations

Default behavior (hide code, show output):

````
markdown

```python {.execute}
print("Hello, World!")
```
````

Show both code and output:

````
markdown

```python {.execute .show-code}
print("Hello, World!")
```
````

Show code only (no output):

````
markdown

```python {.execute .show-code .hide-output}
x = 5 + 3
```
````

Show output only (hide code):

````
markdown

```python {.execute}
print("Result:", 42)
```
````

2.18.5 Simple Print Example

Output only (default):

````
markdown

```python {.execute}
print("Hello, World!")
```
````

```
output

Hello, World!
```

Show both code and output:

````
markdown

```python {.execute .show-code}
print("Hello, World!")
```
````

```
python

print("Hello, World!")
```

```
output

Hello, World!
```

2.18.6 Calculations and Data Processing

Mathematical calculations with NumPy:

````
markdown

```python {.execute .show-code}
import numpy as np
result = np.sum([1, 2, 3, 4, 5])
print(f"Sum of 1 to 5: {result}")
```
````

```
python

import numpy as np
result = np.sum([1, 2, 3, 4, 5])
print(f"Sum of 1 to 5: {result}")
```

```
output

Sum of 1 to 5: 15
```

Multiple lines of output:

````
markdown

```python {.execute .show-code}
for i in range(5):
    print(f"Count: {i}")
```
````

```
python

for i in range(5):
    print(f"Count: {i}")
```

```
output

Count: 0
Count: 1
Count: 2
Count: 3
Count: 4
```

2.18.7 Matplotlib Plots

When your code uses matplotlib, the plot is automatically saved as a PDF and embedded in the document.

Simple plot example (plot only):

````
markdown

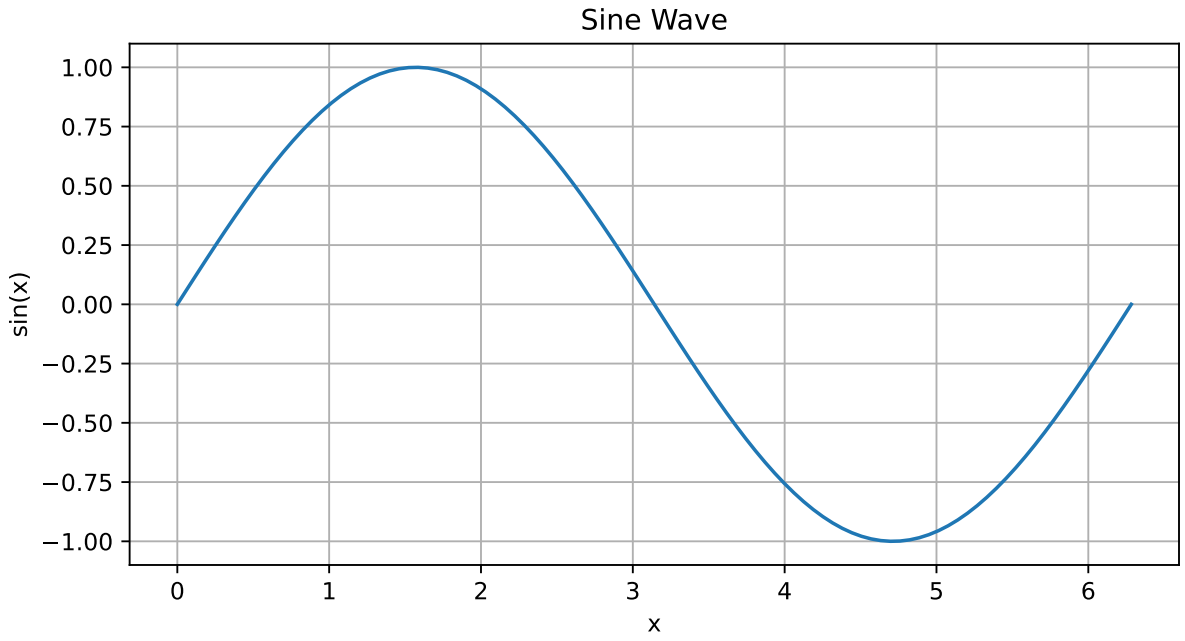
```python {.execute}
import numpy as np
import matplotlib.pyplot as plt

x = np.linspace(0, 2 * np.pi, 100)
```
````

markdown

```
y = np.sin(x)

plt.figure(figsize=(8, 4))
plt.plot(x, y)
plt.title('Sine Wave')
plt.xlabel('x')
plt.ylabel('sin(x)')
plt.grid(True)
plt.show()
```



Polar plot with code shown:

markdown

```
```python {.execute .show-code}
import numpy as np
import matplotlib.pyplot as plt

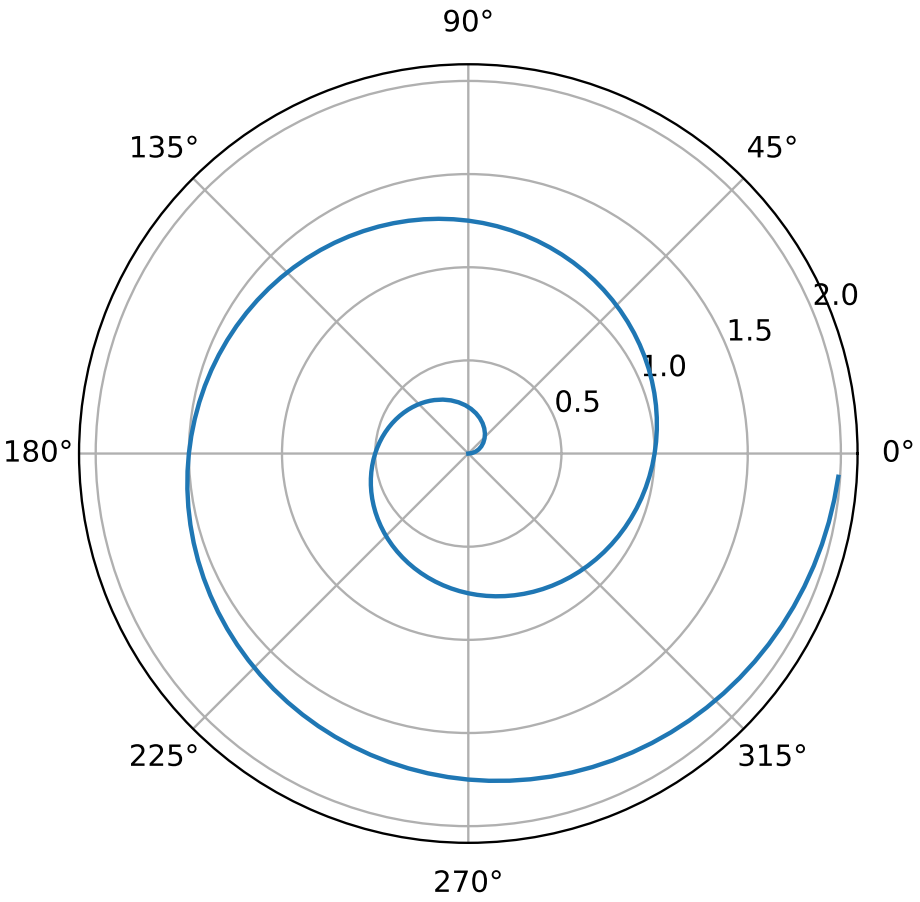
r = np.arange(0, 2, 0.01)
theta = 2 * np.pi * r
fig, ax = plt.subplots(
  subplot_kw = {'projection': 'polar'}
)
ax.plot(theta, r)
ax.set_rticks([0.5, 1, 1.5, 2])
ax.grid(True)
plt.show()
```
```

python

```
import numpy as np
import matplotlib.pyplot as plt

r = np.arange(0, 2, 0.01)
```

```python
theta = 2 * np.pi * r
fig, ax = plt.subplots(
  subplot_kw = {'projection': 'polar'}
)
ax.plot(theta, r)
ax.set_rticks([0.5, 1, 1.5, 2])
ax.grid(True)
plt.show()
```



Multiple subplots:

```markdown
```python {.execute}
import numpy as np
import matplotlib.pyplot as plt

fig, axes = plt.subplots(2, 2, figsize=(10, 8))
x = np.linspace(0, 2 * np.pi, 100)

axes[0, 0].plot(x, np.sin(x))
axes[0, 0].set_title('Sine')

axes[0, 1].plot(x, np.cos(x))
axes[0, 1].set_title('Cosine')

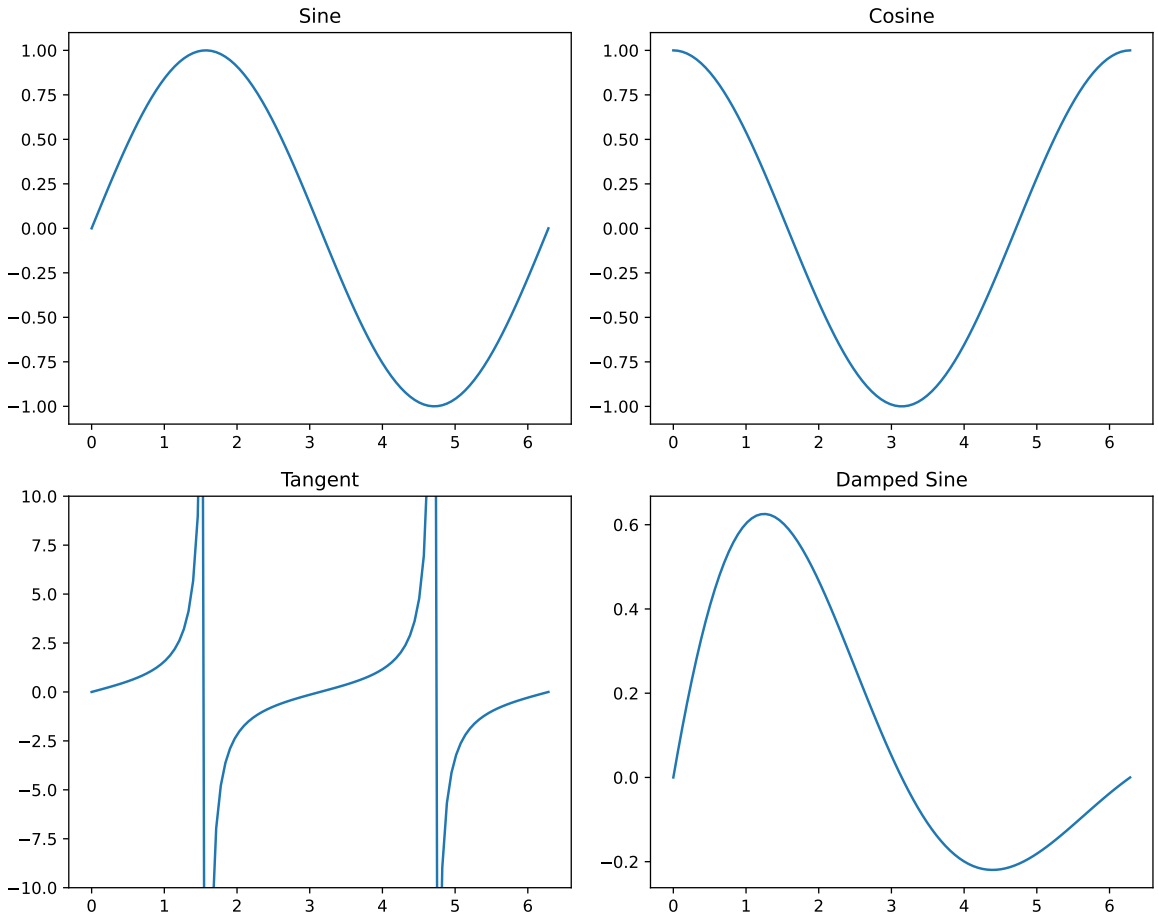
axes[1, 0].plot(x, np.tan(x))
```
```

```
markdown

axes[1, 0].set_title('Tangent')
axes[1, 0].set_ylim(-10, 10)

axes[1, 1].plot(x, np.exp(-x/3) * np.sin(x))
axes[1, 1].set_title('Damped Sine')

plt.tight_layout()
plt.show()
```



2.18.8 Features and Limitations

Features:

- Automatic execution during document build
- Output captured and displayed in `output` blocks
- Matplotlib plots saved as high-quality PDFs
- Flexible visibility control with `.show-code`, `.show-output`, `.hide-code`, `.hide-output`
- Support for all Python libraries (NumPy, Pandas, etc.)
- Error messages displayed if execution fails
- 30-second timeout to prevent hanging

Limitations:

- Code runs in isolated subprocess (no shared state between blocks)

- Interactive plots (`plt.show()`) are automatically converted to saved PDFs
- Execution happens during build time (not runtime)
- Requires Python and necessary libraries installed on your system

Best Practices:

1. Use `.show-code` when teaching or documenting code
2. Omit `.show-code` for cleaner documents where code isn't relevant
3. Keep code blocks independent (they don't share variables)
4. Import all necessary libraries within each code block
5. Use `plt.figure(figsize=(width, height))` to control plot dimensions

2.19 Additional Features

2.19.1 Language Normalization

The following language identifiers are automatically converted for syntax highlighting:

- `jsonc` → `json`
- `tsx` → `typescript`
- `jsx` → `javascript`
- `vue` → `html`
- `svelte` → `html`
- `astro` → `html`

2.19.2 Automatic Processing

- **Image Asset Copying**: All referenced images are automatically copied to build directory
- **Math Protection**: Math expressions are protected during text processing
- **Table Auto-fit**: Tables automatically adjust to fit page width
- **Image Auto-resize**: Images automatically scale to fit page width while maintaining aspect ratio

2.20 Document Metadata (JSON)

Document metadata is configured in a separate JSON file:

```json
{
  "title": "Document Title",
  "subtitle": "Course Name",
  "submittedto": "Professor Name",
  "university": "University Name",
  "department": "Department Name",
  "date": "January 1, 2024",
  "submittedby": [
    {
      "name": "Student Name",
      "roll": "Registration Number"
    }
  ],
  "titleTemplate": 1,
  "enableContentPage": false,
  "enablePageCredits": false,
  "moveFootnotesToEnd": false,
  "enableThatsAllPage": true
}
```

2.20.1 Title Template Modes

The `titleTemplate` setting controls how the title page is displayed:

- `0` : No title (disabled)
- `1` : Full university title page with logo (default) - Good for assignments and reports
- `2` : Title header above content - Good for notes
- `3` : Title on separate page - Good for when the contents are enabled

Controls document structure, metadata, and optional pages.