AMatriz Markdown 101

Learning Markdown Through a Deterministic Document System

Document type: user guide, tutorial, and live system demonstration
Theme variants: amatriz.css and amatriz-print-white.css
Objective: learn Markdown while producing clean, reliable PDF output

Table of Contents Requirements:

- Must be uppercase → [TOC]
- Must NOT be indented
- Must have one blank line after
- Must be in render mode (Ctrl + /)

Result: TOC auto-generates and displays correctly

---

1. What This Document Is

This is not just a guide.

This file teaches Markdown by being Markdown, while simultaneously demonstrating how AMatriz renders documents across:

- screen reading (dark theme)
- PDF export (dark or white)
- long-form technical content

By the end of this document, you should be able to:

- write structured Markdown confidently
- produce clean PDFs without manual layout control
- trust the system to handle pagination, tables, and code

---

2. The Core Principle

Traditional workflow:

Markdown → adjust formatting → add page breaks → export → fix layout → repeat

AMatriz workflow:

Markdown → export → done

There are no manual page breaks required.
Manual page breaks are a `failure mode` in structured documents.

The system controls:

- pagination
- code splitting
- table behaviour
- visual hierarchy

---

3. Basic Markdown Writing

This is a paragraph. It is plain Markdown.

It will render:

- green on black in amatriz.css
- black on white in amatriz-print-white.css

3.1 Emphasis

- **bold text**
- *italic text*
- ***bold italic***
- ~~strikethrough~~

3.2 Inline Code

Use backticks:

`npm install`
`git status`
`@media print`

Inline code is visibly separated and prints cleanly.

---

4. Headings and Structure

Markdown uses # for headings.

One # = H1 Title (used once)

Two ## H2 Major Section

Three ### H3 Subsection

Four ##### H4 Detail

Five ##### H5 Minor

Six ###### H6

Do not skip levels, do not use more than 6 levels.

AMatriz ensures:

- headings stay attached to their content
- no headings are stranded
- structure is preserved in PDF

EVERY SECTION MUST FOLLOW:

Heading
(blank line)
Content

Between blocks:
Content
(blank line)
Next block

Rule:
No stacked elements without spacing.

Text on one line then --- on the line below means the text is a heading.

---

5. Lists

5.1 Unordered Lists

Use - for main items and then indent the dash with 2 spaces for nested items.
All rules must be consistent throughout the document.

- First item
- Second item
  - Nested item
  - Another nested item
- Third item

5.2 Ordered Lists

1. First step
2. Second step
3. Third step

5.3 Task Lists

Use opening [and closing] brackets for check lists, with x marking a filled item.

- [x] Completed task
- [ ] Pending task

---

6. Blockquotes

> This is a blockquote
> used for notes and references

AMatriz ensures:

- clean wrapping across pages
- clear hierarchy

---

7. Code Blocks

Code blocks are critical in technical documents.

7.1 Example

Use ONLY indentation-based code blocks:

```
const system = "AMatriz";
function exportDocument() {
    return "clean PDF";
}
```

AMatriz guarantees:

- code blocks can span multiple pages
- no orphan lines (e.g. single closing brace)

- readable background and borders

7.2 How Code Blocks Actually Work

Markdown treats indentation as structure.

Important rules:

- 4 leading spaces create a code block
- 0–3 spaces = normal text
- Tab adds indentation
- Shift + Tab removes indentation

Example:

This is normal text (no leading spaces)

```
This is code (4 spaces before the line)
```

If your text turns darker green or gains line numbers, Typora recognises it as code.

If formatting breaks:

- remove accidental indentation
- use Shift + Tab
- ensure one blank line before and after code blocks

Indentation is one of the most important Markdown rules.

Understanding it removes most formatting problems.

Indentation is not formatting.
It is a `structural rule`.

7.3 Inline Code Consistency

Always use backticks:

Correct:
`npm install`

Incorrect:
npm install

8. Semantic Annotations (Red and Blue Pills)

AMatriz provides built-in semantic annotations using red and blue inline labels.

These are used to highlight meaning within sentences without affecting layout or structure.

- Red is used for critical rules, constraints, or failure conditions
- Blue is used for optional paths, alternatives, or fallback behaviour

These annotations are intentional and should be used where emphasis is required.

To use these annotations, write inline HTML directly in Markdown:

```
<span class="pill-red">critical rule</span>
<span class="pill-blue">optional fallback</span>
```

These must be written exactly as shown.

> ⓘ Note
>
> **Typora Platform Rendering Behavior**
>
> 1. **Class-Based Pills** (`class="pill-red"` / `class="pill-blue"`):
>   - **On Screen:** Render as **normal text** (Typora's live preview editor filters out custom `class` attributes to avoid theme conflicts).
>   - **In PDF Export:** Render as **colored red and blue pills** (with borders and backgrounds) because Typora's PDF export engine preserves custom HTML classes.
> 2. **Inline Style Colors** (`style="color: red;"` / `style="color: skyblue;"`):
>   - **On Screen & In PDF:** Render as **colored text** in both views because Typora preserves standard inline style attributes in live preview.
>   - **Syntax:**
>   `<span style="color: red;">critical rule</span>`
>   `<span style="color: skyblue;">optional fallback</span>`

Example usage (Class-based - renders as colored pills in PDF only):

Indentation is not formatting.
It is a `structural rule`.

Unicode math can be used as a `fallback` when LaTeX is not available.

Manual page breaks are a `failure mode` in structured documents.

Example usage (Style-based - renders as colored text on screen and in PDF):

Indentation is not formatting.
It is a structural rule.

Unicode math can be used as a fallback when LaTeX is not available.

Manual page breaks are a failure mode in structured documents.

Usage rules:

- Use pills inline within sentences only
- Do not use pills in headings
- Do not place pills on their own line
- Use them sparingly for emphasis (1–2 per section maximum)
- Only use them where meaning needs to be clarified

---

9. Tables

Tables are automatically handled.

Feature	Behaviour	Result
Header	Repeats across pages	Clean continuation
Cells	Wrap content	No clipping
Borders	Preserved	Full visibility

Large tables will:

- split cleanly across pages
- repeat headers automatically

---

10. Links

https://typora.io

Links remain readable in both themes.

---

11. Math

In Typora:

Preferences → Markdown

Enable:

- Inline Math ✅
- LaTeX Math Delimiter ✅
- Code Block Math ✅

Then:
→ Restart Typora

Inline:
$\int_0^1 x^2\,dx = \frac{1}{3}$

Block:

$$\int_0^1 x^2\,dx = \frac{1}{3}$$

Unicode math acts as a `fallback` when LaTeX rendering is not available.

Matrix:

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \cdot \begin{bmatrix} 5 \\ 6 \end{bmatrix} = \begin{bmatrix} 17 \\ 39 \end{bmatrix}$$

Rule:
Always include blank line ABOVE and BELOW block math.

AMatriz ensures:

- equations render cleanly
- spacing remains consistent
- layout does not break

---

12. Long-Form Content

AMatriz is designed for:

- engineering documents

- specifications
- reports

Long paragraphs:

- wrap naturally
- avoid orphan lines
- maintain readability

---

13. Print Behaviour

Dark theme:

- full dark background
- green text

Print white:

- white background
- black text

Both use the same Markdown source.

---

14. What You Do Not Need

You do not need:

- manual page breaks
- layout hacks
- export adjustments

---

15. Correct Workflow

1. Write Markdown
2. Structure content
3. Select theme
4. Export PDF

---

16. Final Principle

Markdown defines structure.
AMatriz defines behaviour.

---

17. Conclusion

Markdown is often treated as a lightweight writing tool.

AMatriz transforms it into something far more reliable:
a structured, predictable document system.

Through this file, you have not only learned how to write Markdown,
but how to control how documents behave when rendered and exported.

You now understand:

- how structure defines meaning
- how indentation controls behaviour
- how code, tables, and math render consistently
- how to produce clean PDFs without manual intervention

The result is a different workflow.

Instead of adjusting layout after writing,
you write with structure, and the system preserves it.

That is the shift.

Markdown defines the intent.
AMatriz guarantees the outcome.

From this point forward, documents are no longer formatted.
They are **engineered**.