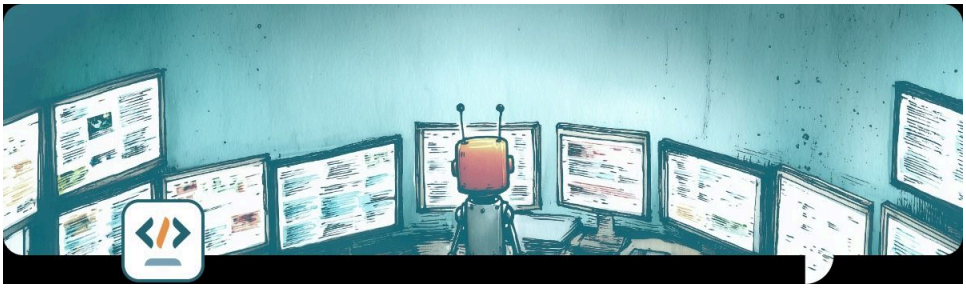
The Power of Markup Languages

Speaking AI's Language Through Structured Formatting



Imagine reading a document without pauses, punctuation, or emphasis; just an endless stream of words. That's probably how AI feels when forced to read unstructured prompts. If you had to endure that day in and day out, you'd hallucinate too. No wonder unstructured prompts produce such wildly unpredictable results. Brilliant insights one minute, complete nonsense the next. When everything runs together in an endless wall of text, no amount of training can help an AI distinguish your instructions from your examples from you banging on your keyboard in frustration.

Markup Languages You Should Be Using

Structure transforms chaotic AI interactions into predictable workflows. It's like adding paragraphs, headings, and emphasis to text, markup languages give AI clear signals about relationships, hierarchies, and data types.

Markdown: Lightweight Formatting

I began using Markdown several years ago with Bear before switching to Obsidian. Since then, many apps I regularly use have adopted Markdown alongside their traditional formatting tools. For all my structured writing needs, this language has become my go-to solution.

Why Markdown

Content-focused workflows thrive on minimal syntax. Community testing shows Markdown is approximately 15% more token efficient than JSON [1], making it the practical choice for most AI interactions.

Unstructured prompt:

```
Write a social media campaign for a sustainable fashion brand targeting Gen Z
that includes 5 posts with hooks, value props, and CTAs focusing on recycled
materials
```

Structured with Markdown:

```
markdown# Social Media Campaign

## Brand Context
- **Industry**: Sustainable fashion
- **Target**: Gen Z
- **Focus**: Recycled materials

## Campaign Requirements
Generate 5 Instagram posts with:
1. Attention-grabbing hook
2. Value proposition
3. Clear CTA
4. Brand voice: authentic, eco-conscious

## Example Tone
"Your style shouldn't cost the Earth 🌍"
```

The structured version consistently produces focused, on-brand content because AI can clearly distinguish between context, requirements, and examples. The unstructured prompt often misses key requirements or tone.

Key Strengths:

- Maximizes token efficiency (~15% better than JSON) [1] [2]
- Provides clear hierarchy through headers and nested lists
- Supports basic formatting and online basics like links
- Easy to learn, easy to read

JSON: Data Precision & Process

At Amazon Web Services (AWS) I picked up JSON basics. When Claude's documentation suggested XML [3], I naturally experimented with JSON too. I use this for programmatic information or light processes (and yes, ChatGPT still fixes my formatting disasters).

Why JSON: When precision matters more than readability, JSON delivers. The research shows JSON excels in hierarchical representation and is particularly effective for tasks requiring detailed structured outputs, though with higher token usage [4].

Example - Managing Design Tokens:

```
json{
  "design_tokens": {
    "colors": {
      "primary": {
```

```
        "value": "#2563EB",
        "contrast_ratio": 7.5,
        "usage": ["buttons", "links", "focus_states"]
      },
      "error": {
        "value": "#DC2626",
        "contrast_ratio": 4.5,
        "usage": ["error_messages", "validation"]
      }
    },
    "spacing": {
      "unit": 8,
      "scale": [0.5, 1, 1.5, 2, 3, 4, 6, 8, 12, 16]
    }
  },
  "ai_instructions": {
    "maintain_wcag_aa": true,
    "preserve_brand_colors": true,
    "suggest_alternatives": false
  }
}
```

This structure ensures AI understands exact color values, accessibility requirements, and constraints. Pure text would lose this precision.

Key Strengths:

- Enforces strict data typing (strings, numbers, booleans, arrays)
- Supports deep nesting for complex data structures
- Provides universal interoperability across programming languages
- Enables validation through JSON schemas
- Balances structure and readability

> **Note**: JSON has limitations with number precision, as most parsers automatically convert numbers to floating point representations, meaning larger values lose precision [5].

XML: Defined Structures with Validation

I've been aware of XML for as long as I can remember, though I never had a real use for it until working with AI. Now, when JSON fails or I'm employing a multi-step process I sacrifice my tokens to the powers of verbosity and put the workhorse to work.

Why XML works: Regulated environments and enterprise systems often require comprehensive validation and audit trails. XML delivers this at the cost of verbosity.

Example - Design Decision Documentation:

```
xml<design_decision>
  <metadata>
    <project>Dashboard Redesign</project>
    <date>2025-05-13</date>
```

```
    <stakeholders>
      <stakeholder role="product_manager">Sarah Chen</stakeholder>
      <stakeholder role="lead_designer">Marcus Williams</stakeholder>
    </stakeholders>
  </metadata>
  <decision type="critical">
    <description>Switch from tabbed to single-page layout</description>
    <rationale>
      <point priority="high">User testing showed 68% task completion
improvement</point>
      <point priority="medium">Reduces cognitive load for new users</point>
      <point priority="low">Aligns with mobile-first strategy</point>
    </rationale>
    <impacts>
      <impact area="development" effort="high">Requires component
refactoring</impact>
      <impact area="design" effort="medium">Need new information
hierarchy</impact>
    </impacts>
  </decision>
</design_decision>
```

Key Strengths:

- Provides comprehensive structure with clear relationships
- Reduces data ambiguity through explicit tags
- Enables strict validation against schemas
- Supports attribute-level metadata
- Integrates with enterprise systems

Note: XML's comprehensive tagging system tends to be more verbose than JSON, requiring additional tokens for equivalent data representation.

Choosing the Right Format

The decision flow is straightforward:

1. **Start with Markdown** - It handles 80% of creative AI use cases
2. **Move to JSON** - When you need data types or API integration
3. **Consider XML** - Only for compliance or enterprise requirements

Quick Decision Guide

Use Markdown when:

- Experimenting with prompts
- Creating content
- Token budget matters
- Humans need to read it

Use JSON when:

- Working with APIs
- Managing configurations
- Data types are critical
- Building reusable templates

Use XML when:

- Compliance requires it
- Schemas provide value
- Enterprise systems demand it
- Audit trails are mandatory

Token Economics

Token consumption directly impacts cost:

- **Markdown**: Most efficient, ~15% better than JSON
- **JSON**: Moderate overhead, good balance
- **XML**: Highest cost, 50-100% more than JSON

For high-volume AI usage scenarios, your choice of markup format directly impacts both token consumption, context window, and associated costs if you're working with pay-as-you-go interfaces.

YAML & TOML: Why I Don't Use Them

Simply put, they're overkill. YAML is another language I've encountered at AWS and after reading up on these two it's clear they're overkill for personal use by creatives. The only thing of interest to me was that TOML offers efficiency advantages over YAML, but the latter is almost everywhere.

For content creation and creative projects, Markdown and JSON provide everything needed without unnecessary complexity.

What's Your Language?

These insights come from my hands-on experimentation with AI tools. I began with Markdown and still use it for majority of my AI interactions. The other formats are there when you'll need them.

More on these languages

Ready to implement structured communication with AI? Explore these practical overviews for format-specific guidance:

- Structuring AI Prompts with Markdown
- Structuring AI Prompts with JSON
- Structuring AI Prompts with XML

Which structured communication patterns have you found most effective in your work? Share your experiences in the comments.

References

[1] OpenAI Community. 2024. "Markdown is 15% more token efficient than JSON." OpenAI Developer Forum, June 26, 2024. https://community.openai.com/t/markdown-is-15-more-token-efficient-than-json/841742

[2] Mukherjee, Anupam. 2025. "Boosting AI Performance: The Power of LLM-Friendly Content in Markdown." Webex Developer Blog, March 13, 2025. https://developer.webex.com/blog/boosting-ai-performance-the-power-of-llm-friendly-content-in-markdown

[3] Anthropic. 2024. "Using XML Tags for Structured Prompt Engineering." Anthropic Documentation, May 2024. https://docs.anthropic.com/en/docs/build-with-claude/prompt-engineering/use-xml-tags

[4] Frontiers in AI. 2025. "Enhancing structured data generation with GPT-4o evaluating prompt styles." Frontiers in Artificial Intelligence, March 2025. https://www.frontiersin.org/articles/10.3389/frai.2025.1558938/full

[5] json-everything. 2023. "Numbers Are Numbers, Not Strings." json-everything blog, May 2023. https://blog.json-everything.net/posts/numbers-are-numbers-not-strings/