# Why XML Tags Are So Fundamental to Claude

*And What That Tells Us About Language*

Guillaume Lethuillier
glthr.com

March 1, 2026 · updated November 15, 2026 · part of the series "Delimiters in Languages"

**Abstract**

I read Anthropic's advice to organize prompts with XML tags through the distinction between evaluated instructions and represented material, such as an email to rewrite. My argument is that the important feature is the boundary the tags mark, rather than XML itself. These tags are ordinary text inside a message, distinct from reserved conversation tokens, and their interpretation by the model provides no security boundary.

> **Update (March 2026):** I updated the considerations around the training vs. model inference framework.

> **Update (November 2026):** When I wrote this, the advice was specific to Claude. Google's Gemini prompting guide now recommends XML-style tags too, and OpenAI's GPT-5 prompting guide writes its own sample prompts in them: what looked like a trait of Claude was an early statement of a general need.

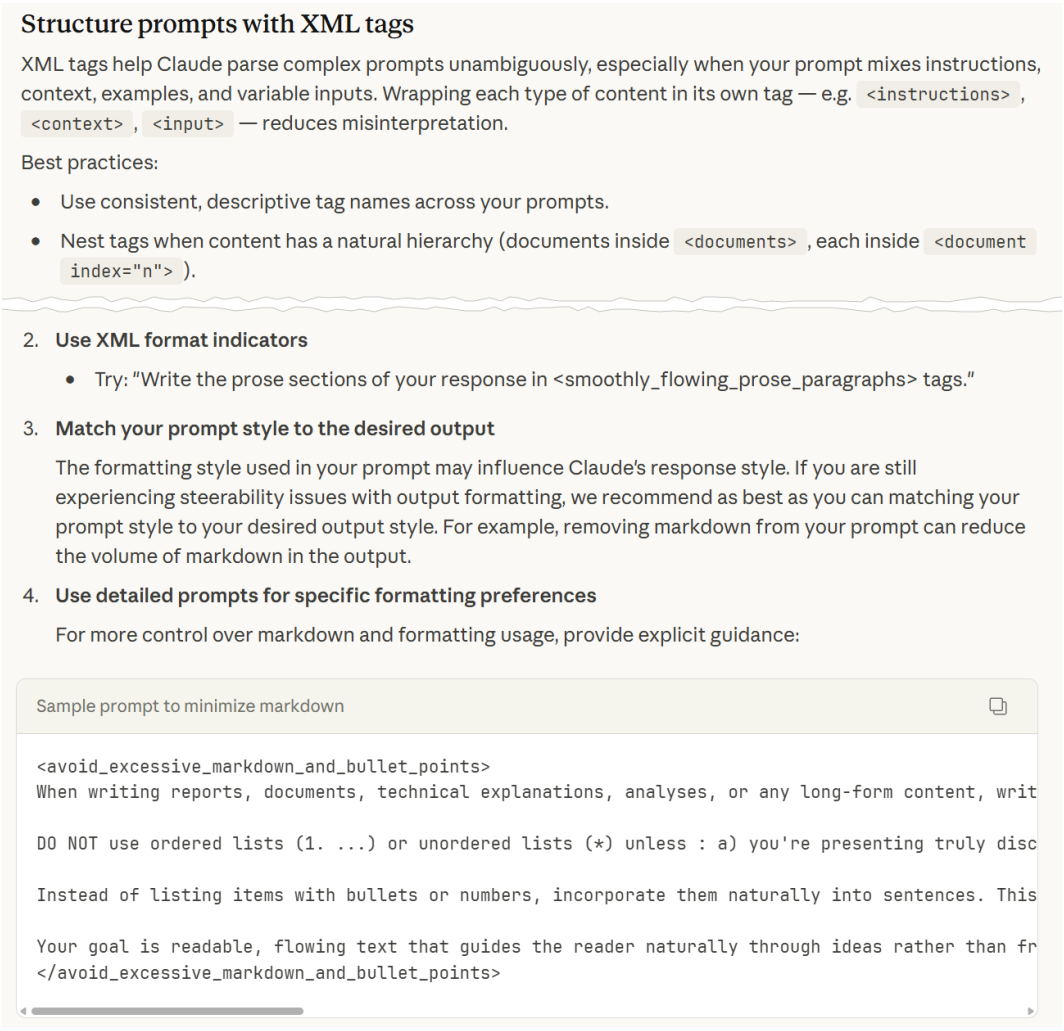
The tour de force of Claude is to have made XML tags first-class citizens. This assertion may seem provocative, but I believe there is something fundamental at play here.

I recommend reading the first article of this series to understand the implication of categorizing XML tags as delimiters in this context.

## What Anthropic Recommends

The Claude API Docs provide practical prompting best practices, designed for developers by outlining specific instructions and clear formatting rules. They return several times to one piece of advice, **which contrasts with the rest of the guide: for the newest models, one of the oldest notations of the web**:

**Cite as:** Guillaume Lethuillier, "Why XML Tags Are So Fundamental to Claude", glthr.com, March 1, 2026. https://glthr.com/xml-fundamental-to-claude



### Structure prompts with XML tags

XML tags help Claude parse complex prompts unambiguously, especially when your prompt mixes instructions, context, examples, and variable inputs. Wrapping each type of content in its own tag — e.g. `<instructions>`, `<context>`, `<input>` — reduces misinterpretation.

Best practices:

- Use consistent, descriptive tag names across your prompts.
- Nest tags when content has a natural hierarchy (documents inside `<documents>`, each inside `<document index="n">`).

2. **Use XML format indicators**
   - Try: "Write the prose sections of your response in <smoothly_flowing_prose_paragraphs> tags."
3. **Match your prompt style to the desired output**

   The formatting style used in your prompt may influence Claude's response style. If you are still experiencing steerability issues with output formatting, we recommend as best as you can matching your prompt style to your desired output style. For example, removing markdown from your prompt can reduce the volume of markdown in the output.
4. **Use detailed prompts for specific formatting preferences**

   For more control over markdown and formatting usage, provide explicit guidance:

Sample prompt to minimize markdown

```
<avoid_excessive_markdown_and_bullet_points>
When writing reports, documents, technical explanations, analyses, or any long-form content, writ

DO NOT use ordered lists (1. ...) or unordered lists (*) unless : a) you're presenting truly disc

Instead of listing items with bullets or numbers, incorporate them naturally into sentences. This

Your goal is readable, flowing text that guides the reader naturally through ideas rather than fr
</avoid_excessive_markdown_and_bullet_points>
```

*Excerpts from Anthropic's documentation as it appeared on March 1, 2026.*

This is not a minor tip: one user reports that structuring prompts with XML can be a transformative experience: "*Here's the simple trick. Instead of just asking Claude stuff like normal, you put your request in special [XML] tags. . . . That's literally it. And the results are so much better.*"

And not only does Claude leverage XML tags in user prompts, but its framework specifically incorporates them as key elements: "*Anthropic heavily uses XML tags in their prompts.*" The system prompts that Anthropic publishes for its own applications show it: they are divided into sections such as `<tone_and_formatting>` and `<user_wellbeing>`.

This is, admittedly, a subjective reading, but I believe **the repurposing of XML, a technology dating back to 1998, is something Claude's makers saw first and said most explicitly: it turns a model into something closer to a genuine language interpreter**.

## Delimiters: First-Order and Second-Order Expressions

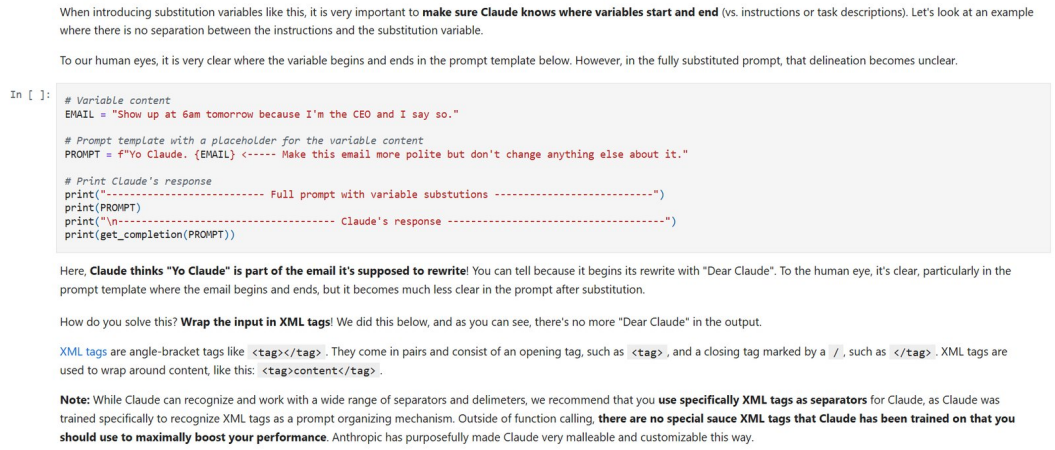
My own research (as a hobbyist) has led me to postulate the existence of a principle I find in every language I have looked at, whether human or artificial. I have observed

this principle at work in diverse contexts: programming languages, bacterial DNA sequences, Homeric verses, and now, seemingly, with Claude. This principle centers on **the necessity for any language (regardless of its form) to possess a mechanism for signaling the transition from first-order to second-order expressions**. I contend that such a mechanism is fundamentally required for information transfer between any two entities; without it, meaningful communication becomes ambiguous.

These transitions are typically indicated by markers or delimiters. In contemporary English, quotation marks serve this purpose. They delineate the shift from direct statement to reported speech, metaphor, or quoted material. These markers operate in pairs: one initiates the transition from first-order to second-order expression, while the other signals a return to the original level of discourse. Furthermore, this nesting can be deeply embedded; we can move from order $n$ to order $n+1$, then to $n+2$, and so on, creating complex layers of meaning.

## The "Yo Claude" Example

To illustrate how these distinctions play out in practice, consider an observation from Anthropic's prompt engineering tutorial, shown here in its Bedrock edition. It serves as a concrete demonstration of how crucial clear delimiters are for ensuring Claude accurately interprets and executes complex prompts:



When introducing substitution variables like this, it is very important to **make sure Claude knows where variables start and end** (vs. instructions or task descriptions). Let's look at an example where there is no separation between the instructions and the substitution variable.

To our human eyes, it is very clear where the variable begins and ends in the prompt template below. However, in the fully substituted prompt, that delineation becomes unclear.

In [ ]:

```
# Variable content
EMAIL = "Show up at 6am tomorrow because I'm the CEO and I say so."

# Prompt template with a placeholder for the variable content
PROMPT = f"Yo Claude. {EMAIL} <----- Make this email more polite but don't change anything else about it."

# Print Claude's response
print("-------------------------- Full prompt with variable substutions --------------------------")
print(PROMPT)
print("\n------------------------------------ Claude's response -------------------------------------")
print(get_completion(PROMPT))
```

Here, **Claude thinks "Yo Claude" is part of the email it's supposed to rewrite**! You can tell because it begins its rewrite with "Dear Claude". To the human eye, it's clear, particularly in the prompt template where the email begins and ends, but it becomes much less clear in the prompt after substitution.

How do you solve this? **Wrap the input in XML tags**! We did this below, and as you can see, there's no more "Dear Claude" in the output.

XML tags are angle-bracket tags like `<tag></tag>`. They come in pairs and consist of an opening tag, such as `<tag>`, and a closing tag marked by a `/`, such as `</tag>`. XML tags are used to wrap around content, like this: `<tag>content</tag>`.

**Note:** While Claude can recognize and work with a wide range of separators and delimeters, we recommend that you **use specifically XML tags as separators** for Claude, as Claude was trained specifically to recognize XML tags as a prompt organizing mechanism. Outside of function calling, **there are no special sauce XML tags that Claude has been trained on that you should use to maximally boost your performance**. Anthropic has purposefully made Claude very malleable and customizable this way.

"*Here, Claude thinks 'Yo Claude' is part of the email it's supposed to rewrite!*" is a remarkably revealing statement. "Yo Claude" is a first-order expression (the user interacting with Claude), the content of the email is a second-order expression (the email the user will address to someone else). And they use XML tags because they need to delimit, they need to enclose the higher-order expression, like we do, in English, when we quote someone using quotation marks (like at the beginning of this paragraph); like Homer did when making heroes talk using formulaic delimiters in Ancient Greek; like bacterial DNA does to store recognition sequences.

A skeptic will ask what happens if the email itself contains `</email>`. Nothing stops the tag from closing early: tags in a prompt have no escaping, and they are a strong hint to the model, not a security boundary (see what happens when a delimiter shows up in plain text).

## Why It Need Not Be XML

In truth, it does not matter that these tags are XML. The tags could have been anything else: brackets, Markdown headings, names of Anthropic's own invention. (They are not the special tokens of a previous article, such as `<|begin_of_text|>`: those frame the conversation itself, and Claude has its own; XML tags are ordinary text typed inside a message.) What matters is what these tags represent. Anthropic says as much in the same tutorial: Claude "*was trained specifically to recognize XML tags as a prompt organizing mechanism,*" yet, outside of function calling, "*there are no special sauce XML tags that Claude has been trained on.*" **What Claude's creators did first, and said most explicitly, was to make their model "aware" of the concept of delimiters, which, at least in my view, is so crucial to the effective processing and communication of information.** And it is precisely this capacity that makes Claude so effective at interpreting layered meaning.

## How to cite this article

> Guillaume Lethuillier, "Why XML Tags Are So Fundamental to Claude", glthr.com, March 1, 2026.
> https://glthr.com/xml-fundamental-to-claude

For BibTeX:

```
@misc{lethuillier2026xml,
  author = {Lethuillier, Guillaume},
  title  = {Why XML Tags Are So Fundamental to Claude},
  year   = {2026},
  month  = mar,
  url    = {https://glthr.com/xml-fundamental-to-claude},
  note   = {glthr.com}
}
```

This document was typeset with LaTeX from the article published at https://glthr.com/xml-fundamental-to-claude, which remains the reference version (as updated on November 15, 2026).