C03 XML Prompting & Semantic Anchors

Design Advanced

Structuring complex prompts for reproducible results

Why XML works

Claude was trained on massive corpora that include XML. When you structure a prompt with tags, you create unambiguous delimiters that separate the instruction, context, reference code, and constraints. The structure can make those roles clearer, but it does not guarantee instruction following or prevent prompt injection.

XML structure is relevant for complex or reusable tasks. For a one-line fix, it is unnecessary noise.

Basic structure

```
<instruction>
Extract the validation logic into a separate module
</instruction>

<context>
Express service, Node.js 20, no external dependencies
</context>

<code_example>
// Existing pattern to preserve
function validateEmail(email: string): boolean {
  return /^[^\s@]+@[^\s@]+\.[^\s@]+$/.test(email)
}
</code_example>

<constraints>
- Do not modify public function signatures
- Maintain compatibility with existing tests
</constraints>

<output>
New file src/validation/index.ts with exports
</output>
```

Reusable tag catalog

Tag	Usage
`<instruction>`	The main task
`<context>`	Stack, system state, history
`<code_example>`	Reference pattern to follow
`<constraints>`	What must not be modified
`<output>`	Expected format and content
`<state>`	Current state (bug, diff, logs)
`<task>` + `<subtask>`	Hierarchical decomposition

Semantic anchors

A semantic anchor names an established method or pattern. It gives Claude a concise reference, but the task still needs project context and a check.

```
Refactor the payment integration using Ports & Adapters.
Keep provider code outside the domain.
Check that the existing payment tests still pass.
```

"Ports & Adapters" names the design approach. The next two sentences say what it means for this task and how to check the change. If Claude applies the term incorrectly, spell out the intended boundary instead of repeating the name. See the guide section on semantic anchors.

Code comments such as `// ANCHOR: auth-flow` are location markers. They can help point to a file region, but they are a different use of the word "anchor."

When to use XML

Case	Use XML?
Simple single-line request	No
Bug to fix in one function	No
Multi-file implementation	Yes
Reusable prompt in a command	Yes
Review with precise criteria	Yes

Integration with CLAUDE.md

You can define the project XML conventions in `CLAUDE.md` to standardize team prompts:

```
## XML Prompt Conventions
Project tags: <api_design>, <accessibility>, <perf_budget>
For features: always include <context> and <constraints>
```

All team members then use the same tag vocabulary, making shared prompts (in `.claude/skills/`) consistent and understandable.