C03 # XML Prompting & Ancres Sémantiques

Conception Advanced

Structurer les prompts complexes pour des résultats reproductibles

Pourquoi XML fonctionne

Claude a été entraîné sur des corpus massifs qui incluent du XML. Quand vous structurez un prompt avec des balises, vous créez des délimiteurs sans ambiguïté qui séparent l'instruction, le contexte, le code de référence et les contraintes. Résultat : Claude interprète chaque section exactement dans le rôle que vous lui assignez, sans devoir inférer ce qui est instruction et ce qui est contexte.

La structure XML est pertinente pour les tâches complexes ou réutilisables. Pour une correction d'une ligne, c'est du bruit inutile.

Structure de base

```
<instruction>
Extraire la logique de validation vers un module
séparé
</instruction>

<context>
Service Express, Node.js 20, pas de dépendances
externes
</context>

<code_example>
// Pattern existant à conserver
function validateEmail(email: string): boolean {
  return /^[^\s@]+@[^\s@]+\.[^\s@]+$/.test(email)
}
</code_example>

<constraints>
- Ne pas modifier les signatures de fonctions
publiques
- Garder la compatibilité avec les tests
existants
</constraints>

<output>
Nouveau fichier src/validation/index.ts avec les
exports
</output>
```

Catalog de tags réutilisables

Tag	Usage
`<instruction>`	La tâche principale
`<context>`	Stack, état du système, historique
`<code_example>`	Pattern de référence à suivre
`<constraints>`	Ce qu'il ne faut pas modifier
`<output>`	Format et contenu attendus
`<state>`	État actuel (bug, diff, logs)
`<task>` + `<subtask>`	Décomposition hiérarchique

Semantic Anchors dans le code

Les anchors sont des marqueurs en commentaire qui permettent à Claude de référencer des zones précises du code sans lire des fichiers entiers.

```
// ANCHOR: auth-flow
export async function authenticate(token: string)
{
  // ...
}
// END-ANCHOR: auth-flow

// ANCHOR: token-validation
function validateJWT(token: string): Claims {
  // ...
}
// END-ANCHOR: token-validation
```

Dans votre prompt : « Modifie uniquement la section ANCHOR: auth-flow pour ajouter la gestion des tokens expirés. » Claude sait exactement où intervenir sans lire tout le fichier.

Quand utiliser XML

Cas	Utiliser XML ?
Requête simple, une ligne	Non
Bug à corriger dans une fonction	Non
Implémentation multi-fichiers	Oui
Prompt réutilisable dans une command	Oui
Review avec critères précis	Oui

Intégration avec CLAUDE.md

Vous pouvez définir les conventions XML du projet dans `CLAUDE.md` pour standardiser les prompts de l'équipe :

```
## Conventions de prompt XML
Tags projet : <api_design>, <accessibility>,
<perf_budget>
Pour les features : toujours inclure <context> et
<constraints>
```

Tous les membres de l'équipe utilisent ensuite le même vocabulaire de tags, ce qui rend les prompts partagés (dans `.claude/commands/`) cohérents et compréhensibles.