M02 Context Management

Methodology Beginner

Managing the context window for efficient sessions

Context indicators

Inspect `/context` for current capacity and the largest consumers.

Compact when earlier details are no longer needed, and start fresh when the task changes. Percentage thresholds can be personal reminders; they do not establish a universal loss of reasoning quality.

Context Sources (by priority)

```
1. ~/.claude/CLAUDE.md       Global rules
2. ./CLAUDE.md               Project rules (root)
3. subfolder/CLAUDE.md       Local rules (cumulative)
4. .claude/rules/*.md        Unscoped rules at startup; paths rules when relevant
5. @file                     Explicit inline import (in session)
```

Management commands

```
/compact      # Summarize earlier context; reduction varies
/clear        # Start a fresh context; saved sessions may be resumed
/usage        # Inspect usage and plan limits
/context      # Detailed context state
```

Adding context intelligently

```
# In the prompt
@src/auth.ts Look at this function

# Via CLAUDE.md (persistent)
Always read package.json before working on deps.

# Via .claude/rules/ (auto-loaded at startup)
Rules scoped by path (paths:) or global
```

Strategies by scenario

Long session (refactor): Make checkpoints at meaningful task boundaries. Save important decisions in `CLAUDE.md`.

Complex debugging: Provide only the relevant files with `@file`. Avoid loading the entire project at once.

Multi-file: Ask Claude to delegate through the Agent tool. `--agent` instead chooses the main session agent.

What consumes the most

Source	Impact
Large files read in full	Very high
Conversation history	High
Verbose command outputs	High
Overly long CLAUDE.md	Medium
Long Claude responses	Medium

Optimizing CLAUDE.md

```
# Good: short and actionable rules
- Always use TypeScript strict mode
- Prefer pnpm over npm

# Bad: long narrative context
We have been working on this project since 2023
and the team decided after many debates to...
```

Common Anti-patterns

Do not paste large JSON or raw logs into the prompt. Use a temporary file and `@file` instead. Do not ignore context warnings: inspect what occupies the context and preserve decisions before compaction.