改めてClaude Codeについて調べてみた

自己紹介

かぐ（Kagu3）

・現役ITエンジニア（業務系システムを主に扱う）
・ITエンジニアキャリア相談・雑談集会　主催
・JUSTゲーム開発集会　主催

Claude Code とは

Anthropic 社が 2025年5月にリリースした
AIエージェント型コーディングツール

- Claude の有料プランで利用可能
- CLIツール
  - VSCode拡張でGUI化可能

AIエージェントとは

ユーザーの目的を達成するために、対話だけでなく様々なツールを自律的に使用してタスクを実行するAIシステム

主な機能

- **情報収集** - Web検索、ファイル読み取り
- **分析・処理** - データ分析、コード実行
- **作成・実行** - ファイル書き込み、コマンド実行
- **計画・判断** - 複数ステップのタスクを自律的に計画・実行

Claude Codeの実装例

Claude Codeでは、以下のツールを通じてAIエージェント機能を実現：

- **Read** - ファイルの読み取り
- **WebSearch** - Web検索
- **Write** - ファイルの書き込み
- **Bash** - シェルコマンドの実行

モデルの種類

モデル	特徴
Claude 4.5 Opus	最高精度 Token 消費が最も多い
Claude 4.5 Sonnet	デフォルトのモデル バランスが良い
Claude 4.5 Haiku	軽量モデル 高速で、Token 消費が少ない

Claude のモデルの特徴

指示追従性が高い

- 指示した通りに動こうとしてくれる

課金プラン

プラン	月額	特徴
Pro	$20	基本プラン
Max x5	$100	Pro の5倍長持ち
Max x20	$200	Pro の20倍長持ち

- ※今回はAPI利用や他プランは扱いません

Claude Code の利用制限

- **5時間ごとの制限**
  - 使いだしてから5時間以内に一定以上利用すると制限で利用不可
- **1週間ごとの制限**
  - 1週間以内に一定以上利用すると制限で利用不可

Pro で Claude Code を使った場合

モデルが Sonnet の場合でも

- 2〜3時間作業していると、**5時間ごとの制限にひっかかる**
- 上記を4〜5回すると**週間制限にひっかかる**

コンテキストウィンドウ管理

- **Pro/Max プランでは200kトークンまで保存可能**
- トークン上限に近づくと**古い会話が自動で要約**
- セッションごとに管理され、他の会話には引き継がれない

トークンとは

AI がテキストを処理する際の最小単位

- 英語: 1単語 ≒ 1〜2 トークン
- 日本語: 1文字 ≒ 1〜2 トークン
- 容量（バイト）とは別概念

AI エージェント用のエンジニアリング

プロンプトエンジニアリング

- どう指示をだすかを考える
  - 簡潔・具体的な指示
  - 簡潔な良い例がひとつあるといい

コンテキストエンジニアリング

- どう長期間効率を維持させるかを考える
  - タスク単位でセッションを分ける
  - Claude Codeの機能をうまく使っていく

Claude Code の主な機能

主なコマンド

コマンド	機能
`/clear`	会話やコンテキストを全消去してリセット
`/compact`	会話履歴を要約してコンテキストウィンドウを空ける
`/rewind`	数ステップ前の状態に戻す
`/context`	現在のコンテキスト内容を表示
`/usage`	課金枠の利用状況を確認

CLAUDE.md

いちいちチャットに打ち込まなくても
自動で読み込まれるコンテキスト

基本仕様

- プロジェクト共通で扱うコンテキスト
- ディレクトリ内とその配下のディレクトリに対して適用される

CLAUDE.md - チーム共有と個人設定

- **チーム共有**: `CLAUDE.md`（Git 管理）
- **個人専用**: `CLAUDE.local.md`（Git 除外）

フォルダ単位の設定

- サブディレクトリに `CLAUDE.md` を配置可能
- そのフォルダとその配下でのみ適用される
- 例: `frontend/CLAUDE.md` は frontend/ 内でのみ有効

Rule

特定の場面のみ適用されるコンテキスト

- `paths` でファイルパターンを指定
- 例: 特定拡張子が対象の場合のみ読み込む
  - `paths: src/**/*.py` → Python ファイルのみ

注意

AIを使わなくていいものは、使わないこと
例：リンターや自動フォーマッターでできることをAIにやらせない

Skill

定型作業の手順をプロンプトとしてまとめる機能

- 作業手順をSkillとして定義し、必要に応じてClaudeがそれを読み込んで実行
- 例: 繰り返し使うテキスト整形処理の手順をSkillにまとめる

subagent

特定分野や目的に特化したサブAIを追加できる機能

- メインのClaudeから特定タスクを委譲でき、単体でも呼び出せる
- 使い分けたい場合や、専門的な知識を持たせたい場合に利用
- コンテキストはエージェント単位なので、メイン側のエージェントのコンテキスト節約に使える
- 例: レビュー専門のサブエージェントを用意して、書き込み禁止にしてチェックだけしてもらう

hook

指定したタイミングで任意のコマンドを自動実行する機能

- Claude Codeのセッションやツールをトリガーに、その前後にコマンド等を行うようにする機能
- 例: write後にフォーマット整形を自動化

カスタムコマンド

自分でClaude Code用のコマンドを作れる

- 例：定型作業のプロンプトを一行で実行できるようにする
- ファイル名がコマンド名になる（例: `review.md` → `/review`）

MCP (Model Context Protocol)

外部サービスやツールと連携するためのプロトコル

よく使われる MCP

- **GitHub**: リポジトリ操作、Issue 管理
- **Playwright**: ブラウザ自動操作・E2Eテスト用のMCP
- **Chrome DevTools**: ブラウザのデバッグや自動化用MCP

注意

MCPは入れるだけでコンテキストウィンドウを消費するため、必要なものだけ入れること

Claude Code でできること

各種機能を組み合わせることで、様々な作業を自律的に実行可能

- **GitHub PR レビュー** - PRに対して自動でコードレビューを実施
- **Issue 対応の自動化** - Issueを作成 → コード修正 → PRまで一連の流れを自律実行
- **テストの自動実行と修正** - テスト実行 → エラー検知 → コード修正を繰り返し

このように、複雑なタスクも自律的に完遂できます

使い方のまとめ - プロンプト

人間への指示と同じ感覚で行えばOK

- **具体的であること**
  - 良い例がひとつあれば更に良い
- **簡潔であること**
- **漏れなく、重複がないこと（MECE）**

使い方のまとめ - コンテキスト

- **タスク単位でセッションを分けること**
- **Claude Codeの機能をうまく使うこと**
  - 最初から全て使おうとせず、必要に応じてClaude Codeと一緒に随時追記していけばOK

各機能の比較

機能	用途	形式	settings.json
CLAUDE.md	コンテキスト定義	Markdown	不要
Rule	条件付きコンテキスト	Markdown	不要
Skills	ワークフロー手順化	Markdown	必要
Sub Agent	タスク委譲	Markdown	不要
Custom Commands	定型プロンプト	Markdown	不要
Hook	イベント駆動実行	settings.json	必要
MCP	外部ツール連携	settings.json	必要

まとめ

- Claude Codeは強力なAIコーディングツール
- プロンプトとコンテキストの工夫で効率アップ
- 豊富な機能を必要に応じて活用
- タスク単位でセッションを分けることが重要

ご清聴ありがとうございました