



# 生成 AI の現在地

# 目次

- 01 自己紹介
- 02 生成 AI の数年史
- 03 生成 AI の現在地
- 04 各トピックをどう捉えているか
- 05 Claude Code 作者による Tips
- 06 まとめ

# 01 自己紹介

齋藤 智樹

X @Tomoki\_Saito\_

1996年生まれ、福島県出身

慶應義塾大学 工学部 数理科学科 卒（統計専攻）

大学卒業後そのままフリーランスエンジニアとして  
キャリアをスタート。業務拡大のため、2021年3月に  
ゼノクリース合同会社を設立。

経営者・コンサル・Webエンジニア・研修講師 など



# 01 自己紹介

## どんな強みがある人？

---

- フリーランスからキャリアを始めたこともあり、何でも選ばずやった結果、技術スタックがだいぶ広がった
- もともと教育業界出身なので、eラーニングや登壇などをする機会が多く、どのような伝え方をすると伝わりやすいか、面白いかの感覚がある
- 自分の会社の代表として日々営業も実装もやっているなので、クライアントがAI活用で共通して気にしている部分が何となく見えてきている

# 01 自己紹介

## 今日のゴール

---

生成 AI をすでに活用している方

→ 「生成 AI 関連の各トピックについて、こんな感じの捉え方をしているのか🤔」というサンプルが1つ増える

これから生成 AI 活用しようと考えている方

→ 「生成 AI 関連の各トピックで名前や話題としては知ってたけどこんな感じのものなのか🤔」という全体像が見える

# 02 生成 AI の数年史

2017.06

【論文】Transformer(Attention Is All You Need)が投稿される  
以後の LLM（自己回帰 / 自己注意）系の型ができる

2019.02

OpenAIが GPT-2 を発表

2020.05

【論文】RAG(Retrieval-Augmented Generation)が投稿される  
検索(retrieval)と生成(generation)を統合する設計

2020.05

【論文】GPT-3(Language Models are Few-Shot Learners)が  
投稿される。プロンプトに数例を示すだけでタスクをこなせるように

# 02 生成 AI の数年史

2021.01



OpenAI が DALL·E (テキスト→画像) を公開

2021.06



GitHub が GitHub Copilot の technical preview を発表  
(生成 AI × 開発の大衆化が始まる)

2022.09



OpenAI が Whisper (音声認識) をオープンソース公開

2022.11



OpenAI が ChatGPT を公開

# 02 生成 AI の数年史

2023.03

OpenAI が GPT-4 を公開  
(モデルの進化で実用性がグッと上がる)

2023.03

OpenAI が ChatGPT plugins を発表  
(外部ツール接続の潮流が強まる)

2023.04

Cursor (エディター) が、VS Code をフォークしてその上に  
構築する形 (v0.2.0) でリリース

2023.06

OpenAI が function calling を含むAPI更新を告知  
(「LLM → 関数呼び出し → 実行」が標準化方向へ)

# 02 生成 AI の数年史

2023.12



Google が Gemini 1.0 を発表

2024.02



OpenAI が Sora (動画生成モデル) を発表

2024.02



Google が Gemini 1.5 を発表  
(長文コンテキストが大きく前進)

2024.03



Anthropic が Claude 3 を発表

# 02 生成 AI の数年史

2024.05

OpenAI が GPT-4o を発表  
(マルチモーダルを実用レベルに)

2024.06

Apple が Apple Intelligence を発表  
(OS 組み込みの「端末 × 個人文脈 × 生成モデル」)

2024.11

Anthropic が Model Context Protocol (MCP) を発表  
(ツール・データ接続の標準化へ)

2025.02

Anthropic が Claude Code を発表  
(CLI ツールのコーディングエージェント、定額の利用料)

# 02 生成 AI の数年史

2025.03

OpenAI が GPT-4o の画像生成を発表  
(画像生成が一般ユーザーにも広がる)

2025.05

OpenAI が Codex を発表  
(クラウドで並列に動くソフトウェアエンジニアリング agent)

2025.05

Anthropic が Claude 4 (Opus 4 / Sonnet 4) を発表  
(エージェント & コーディング用途がかなり強くなる)

2025.06

Cursor v1.0 がリリースされる  
(BugBot / Memories / MCP など、機能がだいぶ固まる)

# 02 生成 AI の数年史

2025.08

OpenAI が gpt-oss:20b, gpt-oss:120b を発表  
(ローカルLLMはこれまでに Llama や Gemma などがあるが)

2025.10

Anthropic が Agent Skills を発表  
(エージェントが外部ツール・手順を扱うための枠組みを前に進める)

2025.11

Google が Gemini 3.0 を発表  
(モデル更新の競争がかなり加熱)

2025.12

Linux Foundation : Agentic AI Foundation (AAIF) 設立  
(MCP ,AGENTS.md などの標準化を目指した業界団体)

# 03 生成 AI の現在地

## 2026 年 1 月時点

---



GPT-5.2 / GPT-5.2 Codex / GPT-5.2 Pro



Claude Sonnet 4.5 / Opus 4.5



Gemini 3.0 Flash / 3.0 Pro

各モデルの賢さ（ユーザーの感想やベンチマークのスコアなど）がかなり成熟してきて、エージェントや周辺ツール、標準化の段階に

# 04 各トピックをどう捉えている？

## ここからのセクション

めっちゃん主観で喋ります

# 04 各トピックをどう捉えている？

## 今日のお願い

---

### 「めっちゃ主観で喋ります」について

- 「公式ドキュメントや書籍、生成 AI での要約などで分かること」ではなく、むしろ生の声や見解をお話ししたいなと思っています。（そのため、今後の展望など明確なデータやエビデンスではなく自分自身が考えていることもお話しします）
- 技術スタックの選定と同様に、現場や案件内容によって生成 AI 活用における正解は異なるので、「こんな例もあるんだな」くらいでお聞きいただくと嬉しいです

# 04 各トピックをどう捉えている？

## チャット

---

普段のチャットでは、最上位モデルだけを使い分けている

- ChatGPT (GPT-5.2 Pro) : Web 検索が上手い
  - Claude (Opus 4.5) : 文章の草案を書くのが上手い
  - Gemini (3.0 Pro) : 画像・動画の分析が上手い (マルチモーダル)
- ※ AI アプリを作る際には、コストを抑えるために Haiku 4.5 などを使うこともある

# 04 各トピックをどう捉えている？

## チャット

---

学習データの大きさの競争から、推論 (Reasoning) の競争の時代に

### LLM の相転移

(学習データやパラメータを一定以上増やすと、精度が飛躍的に上がる)

確率的に、「次にどの語を入れるのがもっともらしいか」を計算するだけでなく、推論時に多くの計算資源や計算ステップを入れる

# 04 各トピックをどう捉えている？

## 調べ物は ChatGPT Pro 一択

### GPT-5.2

さまざまな業界にまたがるコーディングやエージェント型タスクに最適なモデルです。

#### 料金

入力：  
\$1.750 / 100万トークン

キャッシュされた入力：  
\$0.175 / 100万トークン

出力：  
\$14.000 / 100万トークン

### GPT-5.2 pro

最も賢く正確なモデル

#### 料金

入力：  
\$21.00 / 100万トークン

キャッシュされた入力：  
-

出力：  
\$168.00 / 100万トークン

**出力**  
**\$168 / 100万トークン**

<https://openai.com/ja-JP/api/pricing/>

# 04 各トピックをどう捉えている？

ChatGPT 5.2 Pro

共有する ユーザーを追加する ... Google Cloud Identity 概要 · 4m 25s

Google の Cloud Identity って何？

Thought for 4m 25s >

Google の Cloud Identity は、ざっくり言うと「**会社用のID基盤（ユーザー/グループ管理）+SSO/MFA+端末管理**」をまとめてやるサービス。Google 管理コンソールで一元管理できるやつ。 [Google Cloud](#)

主にできることはこれ：

- **ユーザーに SSO でアプリへ入らせる（ログイン体験を一本化）** [Google Cloud](#)
- **多要素認証（MFA）などで保護** [Google Cloud](#)
- **エンドポイント管理（端末ポリシー適用）：私物/社給デバイスに制約かける** [Google Cloud](#)

位置づけとしては：

- **Google Workspace**：メール/ドライブ/Meet 等の“グループウェア”込み
- **Cloud Identity**：それらを抜いた **ID管理・アクセス管理・端末管理だけ**（Workspace を買わずにID基盤だけ欲しいときに刺さる） [Google Cloud +1](#)

GCP 目線だと地味に重要で、**Workspace か Cloud Identity をドメインに紐づけると、Google Cloud の「組織（Organization）」リソースが作られる**（組織配下で統制したい会社運用の入口）。 [Google Cloud Do...](#)

あと、**Free と Premium の2エディション**がある。 [Google Cloud Do...](#)

情報源

質問してみましょう

+ Extended thinking

ChatGPT の回答は必ずしも正しいとは限りません。重要な情報は確認するようにしてください。cookie の設定を参照してください。

• Brief explanation of Google Cloud Identity  
Cloud Identity is a Google service for managing users, groups, apps, and devices through Google Admin Console. It offers SSO, MFA, and device policies. It's built into both Google Cloud and Workspace, but can be used standalone if you don't need Workspace services like Gmail. It's different from Google Cloud IAM, which handles permissions, not user management. Also, it's distinct from Identity Platform, which focuses on customer authentication. There are free and premium editions, with the free version offering 50 user licenses.

Thought for 4m 25s 完了

情報源 · 51

[Google Cloud](#)  
**Cloud Identity**  
ID、アクセス、アプリ、エンドポイント管理（IAM / EMM）の統合プラットフォーム。... Google は Gartne...

[Google Cloud](#)  
**Identity & Access management: Authentication with Cloud ...**  
2022年7月25日 — Jul 25, 2022 — Cloud Identity is the identity provider (IdP) for Google Cloud. It also is the...

[Google Cloud Documentation](#)  
**組織リソースを作成、管理する | Resource Manager**  
Google Workspace アカウントまたは Cloud Identity アカウントを作成し、ドメインに関連付けると、組織リ...

[Google Cloud Documentation](#)  
**Editions | Cloud Identity**  
Cloud Identity offers a free subscription option and a premium option with additional enterprise features for...

出力モデルの金額が  
すごく高いだけでなく、  
並列推論を行って  
「良いところ取り」  
をする形なので、調べ物  
をするときはダントツで  
優秀

→自分が気になったこと  
の検証や、クライアント  
からの質問を入れる

# 04 各トピックをどう捉えている？

## チャット

---

とはいえ、ChatGPT Pro（Proに限らず）は変な造語やカッコ付けた言い回しをしてることが多々あるので、

ChatGPT Pro で叩き台と骨格

→ Claude Opus 4.5 でそれを分かりやすい

（自分が読んでも分かる）形に翻訳しつつ、自分で加筆修正

# 04 各トピックをどう捉えている？

## マルチモーダル

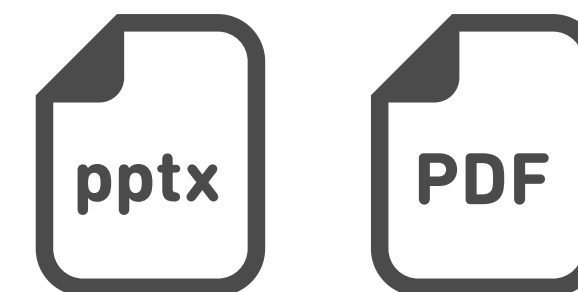
---

単一画像の読み取り精度は上がってきているが、複合ファイルに課題

PDF も、「画像のみの PDF（スキャナーでスキャンしたものとか）」と  
「文字情報が入った PDF（Word, Markdown から作成したものとか）」  
があり、アプローチが異なる。

Word, Excel, PowerPoint などは、Open-XML 形式

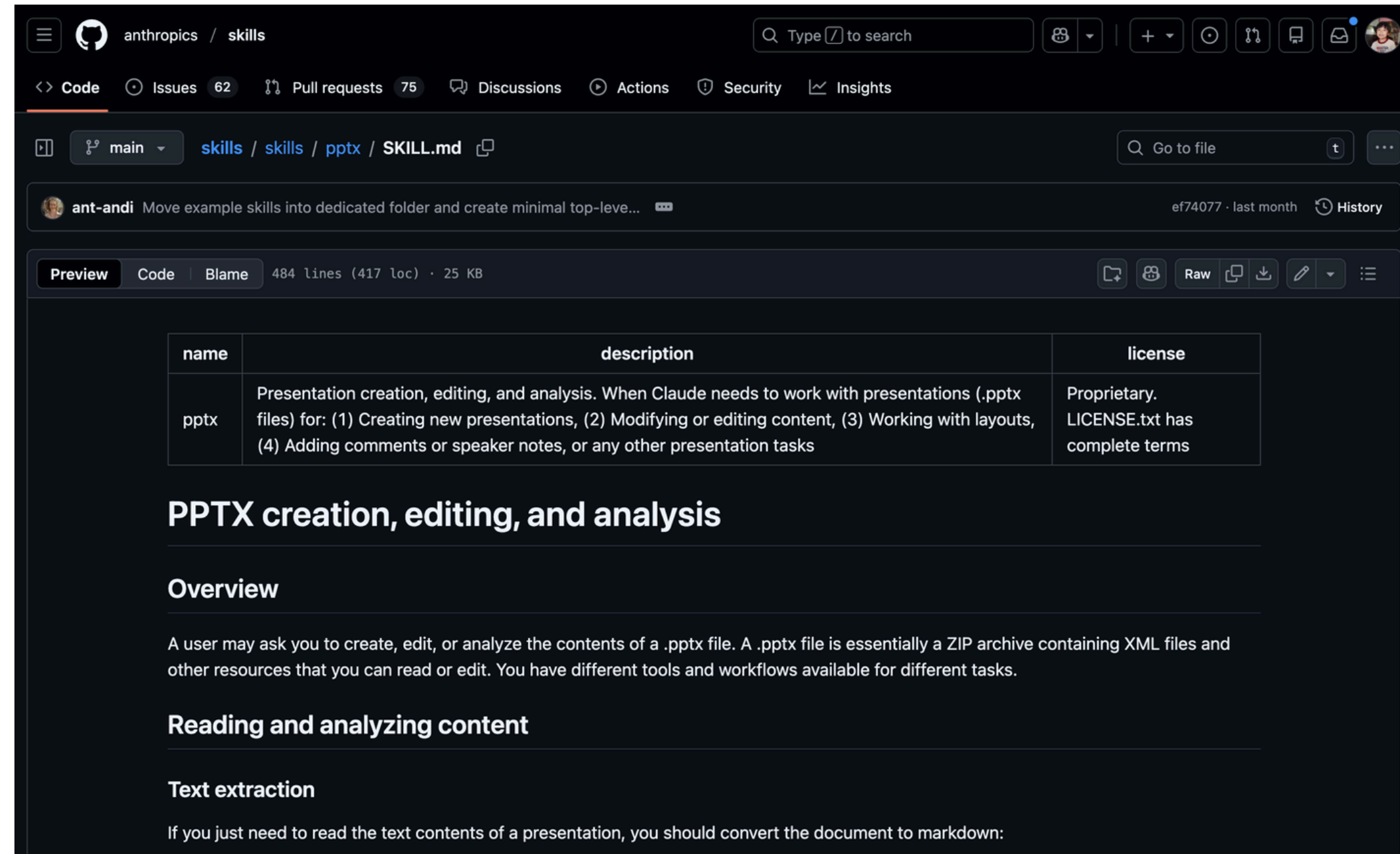
<https://learn.microsoft.com/ja-jp/office/open-xml/open-xml-sdk>



# 04 各トピックをどう捉えている？

## マルチモーダル

Anthropic (Claude)公式の Agent Skillsが参考になるものの、あくまで JS, Python の docx, pptx, xlsx 等のライブラリの使い方などを分かりやすくまとめているだけなので、本格的にはもっとやる必要がある



The screenshot shows a GitHub pull request for the file 'SKILL.md' in the 'anthropics / skills' repository. The pull request is titled 'Move example skills into dedicated folder and create minimal top-level...' and was submitted by user 'ant-andi' last month. The file is 484 lines (417 loc) and 25 KB. The content of the file is a table with columns 'name', 'description', and 'license'. Below the table, there is a section titled 'PPTX creation, editing, and analysis' with an 'Overview' subsection. The overview text states: 'A user may ask you to create, edit, or analyze the contents of a .pptx file. A .pptx file is essentially a ZIP archive containing XML files and other resources that you can read or edit. You have different tools and workflows available for different tasks.' There are also sections for 'Reading and analyzing content' and 'Text extraction'.

name	description	license
pptx	Presentation creation, editing, and analysis. When Claude needs to work with presentations (.pptx files) for: (1) Creating new presentations, (2) Modifying or editing content, (3) Working with layouts, (4) Adding comments or speaker notes, or any other presentation tasks	Proprietary. LICENSE.txt has complete terms

### PPTX creation, editing, and analysis

#### Overview

A user may ask you to create, edit, or analyze the contents of a .pptx file. A .pptx file is essentially a ZIP archive containing XML files and other resources that you can read or edit. You have different tools and workflows available for different tasks.

#### Reading and analyzing content

#### Text extraction

If you just need to read the text contents of a presentation, you should convert the document to markdown:

<https://github.com/anthropics/skills/blob/main/skills/pptx/SKILL.md>

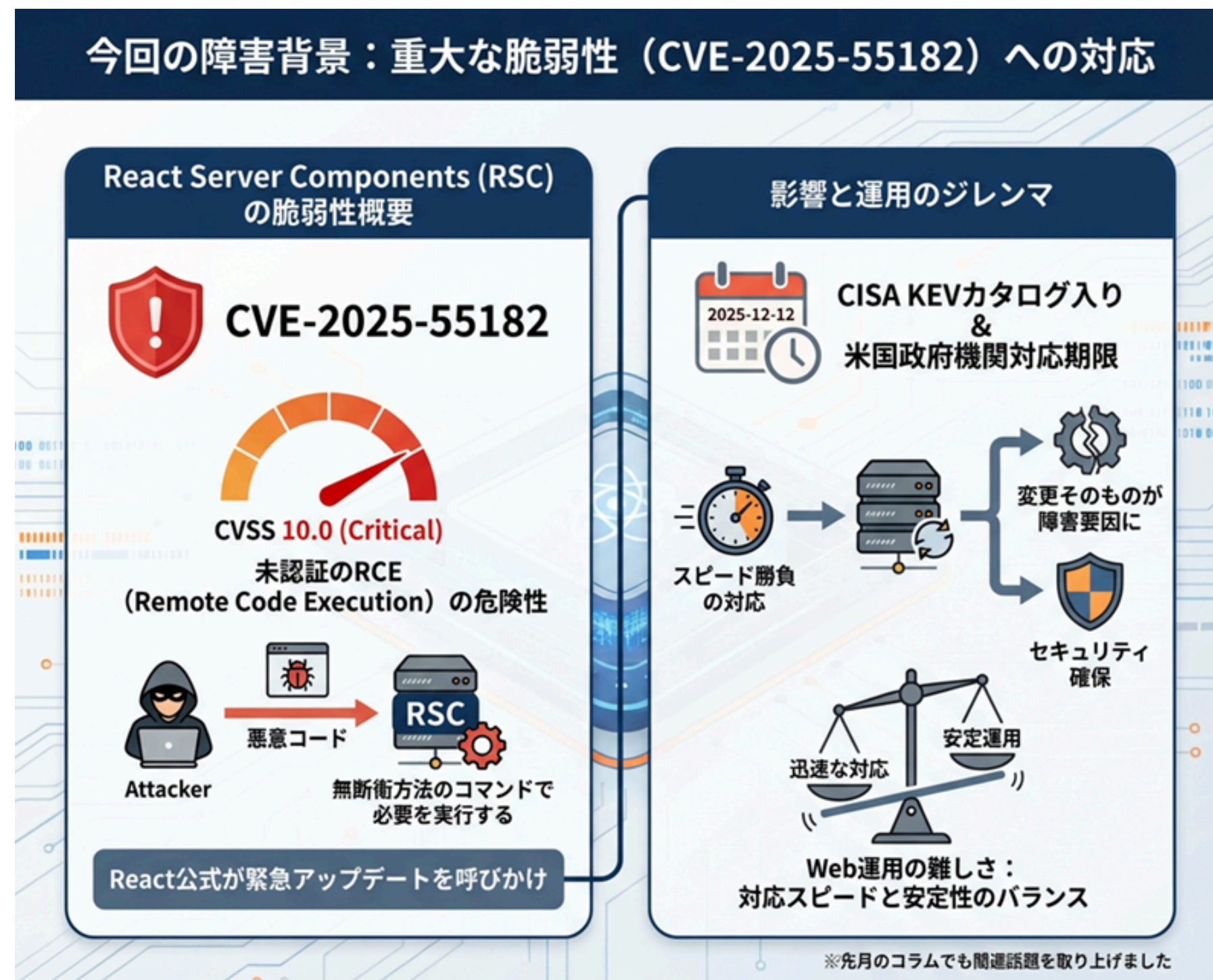
# 04 各トピックをどう捉えている？

## 画像生成

### 主力モデル

- gpt-image-1.5 (OpenAI)
- gemini-3-pro-image-preview (別名 Nano Banana Pro, Google)

文字崩れ問題（日本語と中国語の中間みたいな）も解消してきて、実運用のラインにだいぶ乗ってきた感はある



# 04 各トピックをどう捉えている？

- **高解像度出力:** 1K、2K、4K のビジュアルを生成する機能が組み込まれています。
- **高度なテキスト レンダリング:** インフォグラフィック、メニュー、図表、マーケティング アセット用に、読みやすくスタイリッシュなテキストを生成できます。
- **Google 検索によるグラウンディング:** モデルは、Google 検索をツールとして使用して、事実を確認し、リアルタイムのデータ（現在の天気図、株価チャート、最近のイベントなど）に基づいて画像生成できます。
- **思考モード:** モデルは「思考」プロセスを使用して、複雑なプロンプトを推論します。最終的な高品質の出力を生成する前に、構成を調整するための中間的な「思考画像」（バックエンドで表示されるが、課金されない）を生成します。
- **最大 14 枚の参照画像:** 最大 14 枚の参照画像を組み合わせて最終的な画像を生成できるようになりました。

## 最大 14 枚の参照画像を使用する

Gemini 3 Pro Preview では、最大 14 枚の参照画像を組み合わせることができます。これらの 14 枚の画像には、次のものを含めることができます。

<https://ai.google.dev/gemini-api/docs/image-generation?hl=ja>

# 04 各トピックをどう捉えている？

## 画像生成

---

API でできることの余地や自由度もだいぶ上がってきている

そのため、実際の案件でもどんどん取り入れていける可能性はある。

社会的には、著作権や肖像権、気持ちの問題など、様々な問題を乗り越えながら一般化していきそう。

(最近の例だと、「Grok の水着問題」など)

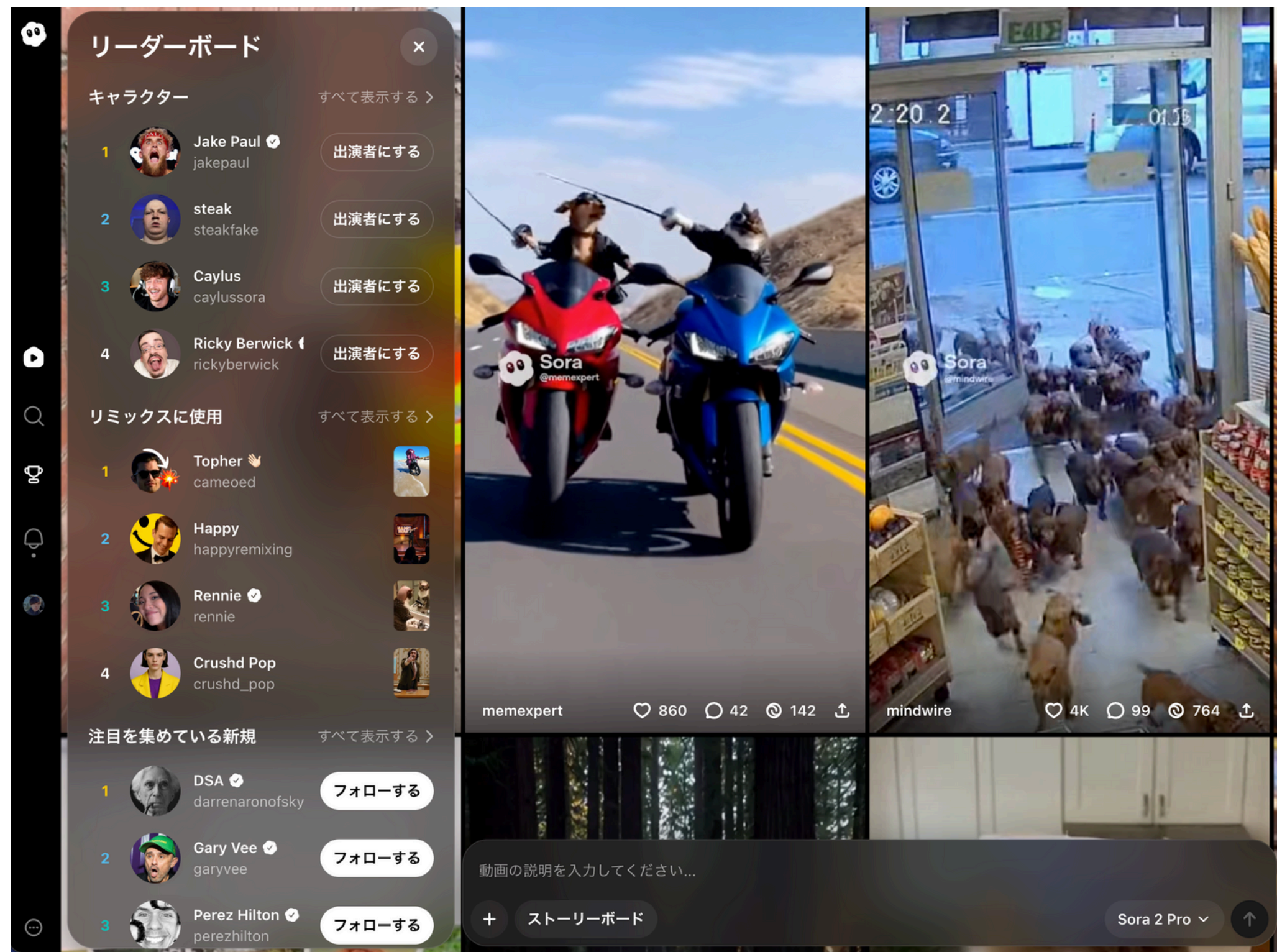
# 04 各トピックをどう捉えている？

## 動画生成

### 主力モデル

- Sora 2 (OpenAI)
- Veo 3.1 (Google)
- LTX-2 (オープンモデル)

例えば Sora 2 (Web 版) では、10s, 15s, 25s の動画を生成できる



# 04 各トピックをどう捉えている？

## 動画生成

---

しかし、API でできることに制限がありすぎて、実運用は難しい

例えば Sora 2 (API 版) では、4s, 8s, 12s の動画しか生成できない

「バラバラに生成して結合する」というパターンだと、意図や作風がバラバラになりすぎて使い物にならない。

また、reference\_image として（実在するかに関わらず）人間の顔を入れると拒否される。（人物の一貫性を担保するのが難しい）

文字崩れ問題の解消もまだまだ

# 04 各トピックをどう捉えている？

## 動画生成

---

法務リスクやブランドイメージとの兼ね合いもあって、

「おもしろ動画制作」の域はまだまだ出なさそう

参考：『JAL高級クレカ「違和感画像」問題 2回の差し替えに至った経緯とは』

<https://news.yahoo.co.jp/expert/articles/31362cf2c44f52ae4f4e820b85f45edb432d002c>

(生成できる動画の秒数、API やプロンプトで指定できるアセットの自由度  
文字崩れ問題の解消など、注視しておくスペックはいくつかある)

# 04 各トピックをどう捉えている？

## エディター・コーディングツール

---

Cursor の上で、Claude Code, Codex を両方使っている

コーディングエージェントに関しては、最もビジネスに寄与していきそうな気配大

「農場を持っているレストランには勝てないよね」問題（定額利用料）

• Claude Code (opus-4.5)



• Codex (gpt-5.2-codex)



• Antigravity (gemini-3.0-pro)



# 04 各トピックをどう捉えている？

## エディター・コーディングツール

---

若干 AGI やシンギュラリティ的なことが起こっている

**AGI 的要素：**

Claude Code などが自律的にコードを読み、テストや検証なども行って、自己改善ループを回す

**シンギュラリティ的要素：**

Claude Code や Codex 自体の開発にも、もちろん大いにこれら自身が寄与している

# 04 各トピックをどう捉えている？

## エディター・コーディングツール

---

これらの要素から、特に変化や発展のスピードが早い

個人的には...

- 「自分が触って見たときの感覚」
- 「中の人（Claude Code の開発者など）」

のどちらか以外は意図的に信じない（話半分で聞く）ようにしている

→ 重要なのは、Tips に理由や根拠があるかどうか

## 04 各トピックをどう捉えている？

### 周辺知識 (MCP / Agent Skills / Slash Command)

---

個人的には、『重要だが重要じゃない』

結局は「プロンプトエンジニアリング」や、「ツールの呼び出し方」のノウハウ

Claude Code は、この部分の仕組みが優れており、Codex や Antigravity と比べても一歩リードしている感はある

(私たちはプロのエンジニアとして、ここは各自で模索していきたい)

# 05 Claude Code 作者による Tips を一緒に見る

## Claude Code 開発チーム Boris Cherny さんのポスト

中の人への情報発信に注目する理由

→ 変化が激しいので、外部の誰かが  
発信した仕様はすでに変わっている  
ことが多い

例：今は Claude Code で “ultrathink”  
と打っても何も変わらない (X より)



<https://x.com/bcherny/status/2007179832300581177>

# 05 Claude Code 作者による Tips を一緒に見る

## 齋藤と見る、作者による Tips

---

先ほどのポストに書いてあった一連の Tips が、個人的には「ほぼ現状の答え」な感じがありました。

Grok で翻訳したものを一緒に見ていきましょう。

# 05 Claude Code 作者による Tips を一緒に見る

🔄 英語からの翻訳 [原文を表示](#)

私はボリスです。私はClaude Codeを作成しました。多くの人からClaude Codeをどう使っているかを聞かれるので、少し私のセットアップを見せたくなりました。

私のセットアップは意外とシンプルかもしれません！ Claude Codeは箱から出したままでも非常にうまく機能するので、私は個人的にあまりカスタマイズしていません。Claude Codeを使う正しい方法は一つではありません：私たちは意図的に、あなたが好きなように使ったり、カスタマイズしたり、ハックしたりできるように構築しています。Claude Codeチームの各メンバーは、それを非常に異なる方法で使っています。

それでは、始めます。

# 05 Claude Code 作者による Tips を一緒に見る

🔄 英語からの翻訳 原文を表示

1/ 私はターミナルで5つのClaudeを並行して実行しています。タブを1-5で番号付けし、システム通知を使ってClaudeが入力必要になったタイミングを知ります [code.claude.com/docs/en/termin...](https://code.claude.com/docs/en/termin...)

この翻訳を評価: 👍 🗨

```
* Fix Forked Skill Progress
1 2 3 4 5
* Fix Forked Skill Progress (node)
└─ Added 1 line, removed 1 line
  12 import type { z } from 'zod/v4'
  13 import type { inputSchema } from './SkillTool.js'
  14 import type { Command } from '../././commands.js'
  15 -import type { ProgressMessage, Message } from '../././types/message.js'
  15 +import type { ProgressMessage } from '../././types/message.js'
  16 import type { Tool } from '../././Tool.js'
  17
  18 type Input = z.infer<typeof inputSchema>

• Now let me run typecheck to verify the changes:

• Bash(bun run typecheck 2>&1 | head -100)
└─ $ bun run build:agent-sdk-typings && tsc
  $ ./scripts/build-ant-sdk-typings.sh
  Generating Agent SDK typings...
  Generating Agent SDK tool typings...

• The typecheck passed. Now let me run the linter on the modified files:

• Bash(bun run lint:file -- src/tools/SkillTool/SkillTool.ts src/tools/SkillTool/UI.tsx src/Tool.ts 2>&1)
└─ $ eslint --fix --no-error-on-unmatched-pattern --cache
  (5s)
  ctrl+b to run in background

* Running typecheck and tests to verify changes... (esc to interrupt · 2m 22s)

>
▶ accept edits on (shift+tab to cycle)
```

```
* fswatch
aude-cli-native-darwin-arm64/package.json
build-ant-native/@anthropic-ai/claude-cli-native/package.json
Build completed successfully!
[]

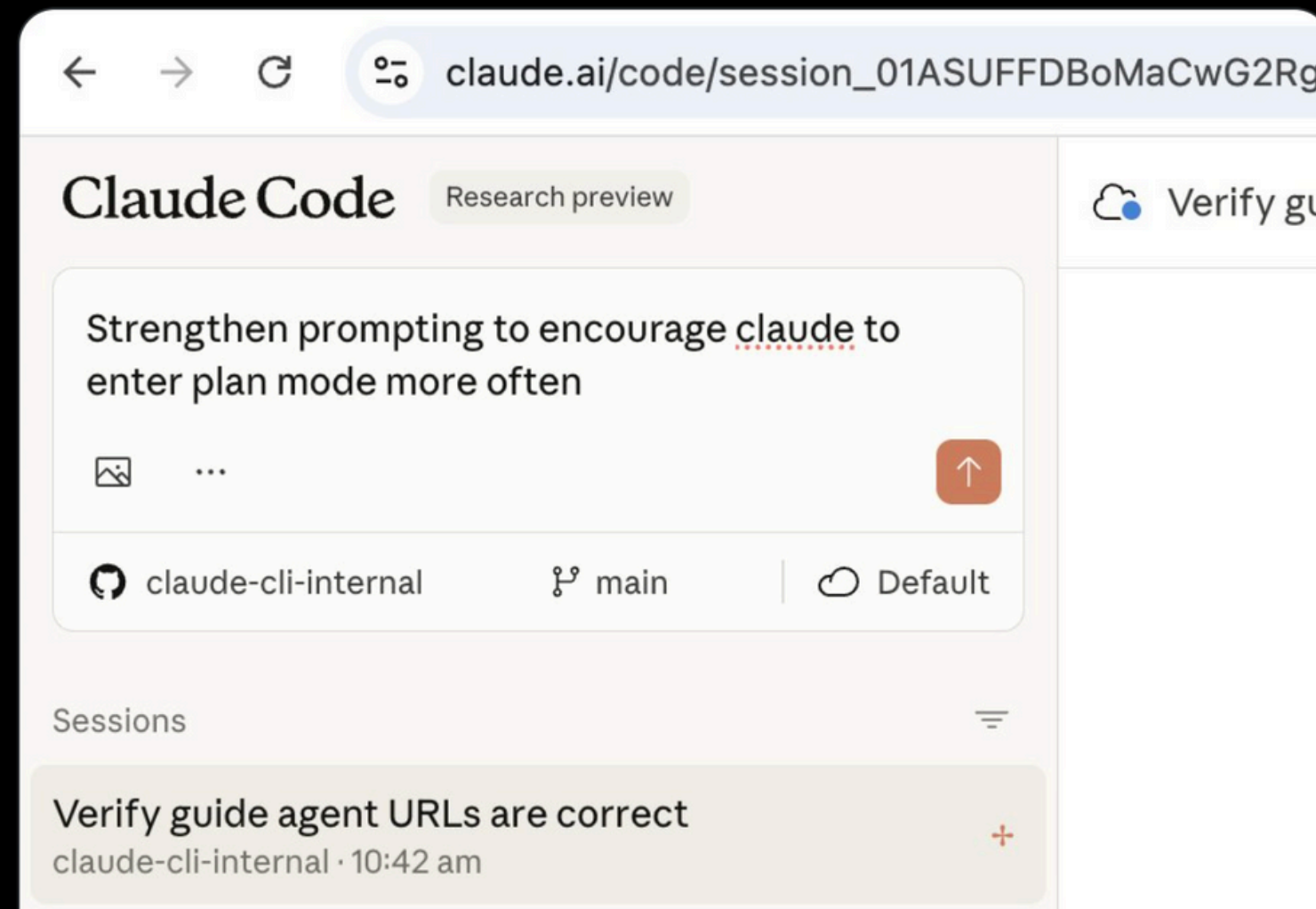
* Math Problem (-zsh)
claude-cli $
```

# 05 Claude Code 作者による Tips を一緒に見る

🔄 英語からの翻訳 原文を表示

2/ 私も [claude.ai/code](https://claude.ai/code) で5~10個のClaudeを並行して実行しています。ローカルのClaudeと一緒にです。ターミナルでコーディングしているとき、私はしばしばローカルのセッションをウェブに引き継ぎます（&を使って）、またはChromeで手動でセッションを開始します。そして時には、--teleportを使って行き来します。また、毎朝と1日のうちに、Claude iOSアプリから電話でいくつかのセッションを開始し、後でそれらをチェックします。

この翻訳を評価: 👍 🗨



# 05 Claude Code 作者による Tips を一緒に見る

🗨️ 英語からの翻訳 [原文を表示](#)

3/ 私はすべてのために思考付きのOpus 4.5を使っています。それは私が今まで使った中で最高のコーディングモデルで、Sonnetより大きくて遅いにもかかわらず、操縦する必要が少なくツール使用が優れているので、結局のところ小さなモデルを使うよりもほぼ常に速いです。

この翻訳を評価: 👍 🗨️

# 05 Claude Code 作者による Tips を一緒に見る

🔄 英語からの翻訳 [原文を表示](#)

4/ 私たちのチームは、Claude Code リポジトリのために1つの [CLAUDE.md](#) を共有しています。私たちはそれを git にチェックインし、チーム全体で週に何度も貢献します。Claude が何か間違ったことをしたのを見かけるたびに、それを [CLAUDE.md](#) に追加します。そうすれば、Claude は次回それをしないように知ることができます。

他のチームはそれぞれ自分の [CLAUDE.md](#) を管理しています。各チームが自分のものを最新の状態に保つのが仕事です。

この翻訳を評価: 👍 🗨️

```
claude-cli $ cat CLAUDE.md
# Development Workflow

**Always use `bun`, not `npm`.**

```sh
# 1. Make changes

# 2. Typecheck (fast)
bun run typecheck
```

# 05 Claude Code 作者による Tips を一緒に見る

🔄 英語からの翻訳 [原文を表示](#)

5/ コードレビュー中に、同僚のPRに @.claude をタグ付けして、[CLAUDE.md](#) に何かを追加することがよくあります。これはPRの一部としてです。私たちはこれのために Claude Code Github action (/install-github-action) を使用しています。これは @danshipper の Compounding Engineering の私たちのバージョンです。

この翻訳を評価: 👍 🗨

**bcherry** 3 days ago

nit: use a string literal, not ts enum

@claude add to CLAUDE.md to never use enums, always prefer literal unions

👍 1 🗨 1

**claude** bot 3 days ago · edited

Claude finished @bcherry's task in 39s — [View job](#)

---

**Adding enum guidance to CLAUDE.md**

- ✓ Read current CLAUDE.md to understand existing guidance
- ✓ Update CLAUDE.md to strengthen "no enums" guidance
- ✓ Commit the change

Updated CLAUDE.md line 36 from:

```
- Prefer `type` over `interface`; avoid `enum` (use string unions)
```

to:

```
- Prefer `type` over `interface`; **never use `enum`** (use string literal unions instead)
```

Committed in [156a458](#).

# 05 Claude Code 作者による Tips を一緒に見る

🔄 英語からの翻訳 [原文を表示](#)

6/ ほとんどのセッションはPlanモード (shift+tabを2回) で始まります。私の目標がPull Requestを書くことなら、Planモードを使い、Claudeと行き来しながらそのプランが気に入るまで調整します。そこから、auto-accept editsモードに切り替え、Claudeは通常1回で完了させられます。良いプランは本当に重要です！

この翻訳を評価: 👍 👎

```
> i want to improve progress notification rendering for skills. can you make it look and feel a bit more like subagent progress? █
```

```
|| plan mode on (shift+tab to cycle)
```

# 05 Claude Code 作者による Tips を一緒に見る

🔄 英語からの翻訳 [原文を表示](#)

7/ 私は、1日に何度も繰り返す「内側ループ」のワークフローごとにスラッシュコマンドを使用しています。これにより、繰り返しのプロンプトを避けられ、Claudeもこれらのワークフローを使えるようになります。コマンドはgitにチェックインされており、`.claude/commands/` に保存されています。

たとえば、Claudeと私は、`/commit-push-pr` というスラッシュコマンドを毎日何十回も使用しています。このコマンドは、インラインのbashを使ってgitのステータスや他のいくつかの情報を事前に計算し、コマンドを素早く実行し、モデルとのやり取りを避けるために使われます。

この翻訳を評価: 👍 👎

```
> /commit-push-pr █
```

```
/commit-push-pr      Commit, push, and open a PR
```

# 05 Claude Code 作者による Tips を一緒に見る

🔄 英語からの翻訳 [原文を表示](#)

8/ 私はいくつかのサブエージェントを日常的に使っています：code-simplifier は Claude が作業を終えた後にコードを簡略化し、verify-app は Claude Code をエンドツーエンドでテストするための詳細な指示を持っています、など。スラッシュコマンドに似ていて、私はサブエージェントを、ほとんどの PR で私がやっている最も一般的なワークフローを自動化するものだと考えています。

この翻訳を評価: 👍 🗨

▽ .claude

▽ agents

- ➡ build-validator.md
- ➡ code-architect.md
- ➡ code-simplifier.md
- ➡ oncall-guide.md
- ➡ verify-app.md

# 05 Claude Code 作者による Tips を一緒に見る

🔄 英語からの翻訳 [原文を表示](#)

9/ 私たちはPostToolUseフックを使用してClaudeのコードをフォーマットします。Claudeは通常、最初からよくフォーマットされたコードを生成しますが、このフックは後でCIでのフォーマットエラーを避けるために最後の10%を処理します。

この翻訳を評価: 👍 🗨

```
"PostToolUse": [  
  {  
    "matcher": "Write|Edit",  
    "hooks": [  
      {  
        "type": "command",  
        "command": "bun run format || true"  
      }  
    ]  
  }  
]
```

# 05 Claude Code 作者による Tips を一緒に見る

🔄 英語からの翻訳 [原文を表示](#)

10/ 私は `--dangerously-skip-permissions` を使いません。代わりに、`/permissions` を使って、私の環境で安全だとわかっている一般的な `bash` コマンドを事前に許可し、不要な権限プロンプトを避けています。これらのほとんどは `.claude/settings.json` にチェックインされており、チームと共有されています。

この翻訳を評価: 👍 🗨

> /permissions

Permissions: **Allow** Ask Deny Workspace (←/→)

Claude Code won't ask before using allowed tools.

🔍 Search...

- ↑ 12. Bash(bq query:\*)
- 13. Bash(bun run build:\*)
- 14. Bash(bun run lint:file:\*)
- 15. Bash(bun run test:\*)
- 16. Bash(bun run test:file:\*)
- 17. Bash(bun run typecheck:\*)
- 18. Bash(bun test:\*)
- 19. Bash(cc:\*)
- 20. Bash(comm:\*)
- > 21. Bash(find:\*)

# 05 Claude Code 作者による Tips を一緒に見る

🔄 英語からの翻訳 [原文を表示](#)

11/ Claude Code は、私のすべてのツールを代わりに使ってくれます。よく検索して Slack に投稿したり（MCP サーバー経由で）、分析の質問に答えるために BigQuery クエリを実行したり（bq CLI を使用して）、Sentry からエラーログを取得したりします。Slack MCP の設定は、.mcp.json にチェックインされており、チームと共有されています。

この翻訳を評価: 👍 🗨

```
claude-cli-2 $ cat .mcp.json
{
  "mcpServers": {
    "slack": {
      "type": "http",
      "url": "https://slack.mcp.anthropic.com/mcp"
    }
  }
}
```

# 05 Claude Code 作者による Tips を一緒に見る

🔄 英語からの翻訳 [原文を表示](#)

12/ 非常に長時間実行されるタスクの場合、私は (a) Claude に完了時にバックグラウンドエージェントでその作業を検証するようプロンプトする、(b) より決定論的にそれを行うためにエージェントの Stop hook を使用する、または (c) ralph-wiggum プラグイン (元々 [@GeoffreyHuntley](#) が考え出したもの) を使用します。また、セッション中のパーミッションプロンプトを避けるために、サンドボックス内で `--permission-mode=dontAsk` または `--dangerously-skip-permissions` のいずれかを使用し、Claude が私にブロックされずに作業を進められるようにします。

[github.com/anthropics/cla...](https://github.com/anthropics/cla...)

この翻訳を評価: 👍 🗨

```
* Reticulating... (1d 2h 47m · ↓ 2.4m tokens · thinking)
```

```
> █
```

# 05 Claude Code 作者による Tips を一緒に見る

🔄 英語からの翻訳 原文を表示

13/ 最後のヒント：Claude Codeから優れた結果を得るために、おそらく最も重要なこと——Claudeにその作業を検証する方法を与えることです。Claudeにそのフィードバックループがあれば、最終結果の品質を2-3倍に高めます。

Claudeは私が[claude.ai/code](https://claude.ai/code)に適用するすべての変更を、Claude Chrome拡張機能を使ってテストします。ブラウザを開き、UIをテストし、コードが動作しUXが良好になるまで反復します。

検証は各ドメインで異なります。bashコマンドを実行するだけの場合もあれば、テストスイートを実行する場合、またはブラウザや電話シミュレーターでアプリをテストする場合があります。これを確実なものにするために投資してください。

[code.claude.com/docs/en/chrome](https://code.claude.com/docs/en/chrome)

この翻訳を評価: 👍 👎

## 06 おわりに

プログラミングは民主化 (Vibe Coding など) したが、  
まだまだプロにしかできないことが多く、代替はされない

---

短期・限られた視点で見ると、ジュニアエンジニアの採用問題や  
受託開発の縮小など暗いニュースも多いが...

むしろ、「エンジニアの構造化・抽象化能力が、ホワイトワーカーの  
仕事全般の領域に広げられる」 ようになったと考えるべき

『ソフトウェア開発の天井は非常に高い』

Y Combinator の Garry Tan による、Anysphere社  
(Cursor 開発元)CEO Michael Truell へのインタビューより

# 06 おわりに

## ゼノクリースと共に進みましょう！

---

クライアント様と共に良いものを創り上げられるよう、  
いつでも万全な準備を整えております。営業や商談の支援  
も承わっております。お気軽にご連絡ください！

Web <https://xenoculis.co.jp>

TEL 03-6403-9271

Email [contact@xenoculis.co.jp](mailto:contact@xenoculis.co.jp)

**F i n .**