2026.08.21 FRI

チートシート

研修後に手元で引くことを想定した3本です。指示の型と直し方、Claude Code の道具の作り方、機密情報の扱い。それぞれ独立して読めます。

CONTENTS

1. バイブコーディング チートシート
2. Claude Code 道具の作り方 チートシート
3. 機密情報の扱い チートシート

バイブコーディング チートシート

2026年8月21日 GenAI活用スキル向上支援 第3回／株式会社ギブリー 安田 光喜

指示の型 4段階

段階	指示のしかた	何が変わるか	どこに残るか
1	思いついたまま話す	動くものは出ます。毎回違うものが出ます	どこにも残りません
2	型に沿って書く	狙ったものが出る確率が上がります	手元のメモ
3	型そのものを道具にする	毎回同じ型で出ます。人に渡せます	`.claude/skills/`

段階	指示のしかた	何が変わるか	どこに残るか
4	道具を並べて流す	手順ごと再現できます。分担できます	Skill ＋ サブエージェント ＋ フック

段階2で書く項目は4つです。何を作るか、誰のためか、何ができたら完成か、何をしてはいけないか。ワークシート②がこの型になっています。

完成の条件

終わらない言い方のまま渡すと、AIはどこかで勝手に手を止めます。判定できる形に書き換えてから渡してください。

終わらない言い方	判定できる言い方
見やすくする	幅1280pxの画面で、縦スクロールなしに全件が見える
使いやすくする	1件追加するのにクリック2回以内で終わる
速くする	100件のデータで、開いてから1秒以内に一覧が出る
きれいなデザインにする	色は DESIGN.md の2色だけを使う。3色目を足さない
エラーに強くする	締切が空欄のまま保存しても画面が白くならず、「締切を入れてください」と出る
スマホでも見られるようにする	画面幅390pxで横スクロールが出ない
データを保存する	ブラウザを閉じて開き直しても、追加した3件が残っている
検索できるようにする	案件名の一部を入れると一致する行だけが残り、0件のときは「該当なし」と出る

右の列は、他人が読んでも同じ判定になります。そこまで書けたら渡してください。

直す指示の3点セット

項目	何を書くか	例
現状	いま画面がどうなっているか	締切の列が日付だけで、あと何日かが分からない
達したい状態	どうなってほしいか	締切の右に「あと3日」と出す。過ぎているものは「2日超過」
制約	触ってほしくない場所	列の並びは変えない。色は増やさない

制約を書かないと、頼んでいない場所まで直されます。3点のうち一番効くのがここです。

スクリーンショット

撮り方は Mac が `Shift + Command + 4`、Windows が `Windows + Shift + S`。どちらも範囲を選んで撮ります。

貼り方は Claude Code の入力欄で `Ctrl + V`。Mac でも `Ctrl + V` です。`Cmd + V` が効くのは iTerm2 だけなので、ここは間違えやすい箇所です。貼れないときは、画像ファイルをウィンドウにドラッグしても入ります。

Anthropic のドキュメントは「Claude works best when images come before text」と書いています。画像を先に貼り、そのあとに1行添えてください。画像だけでは、どこを見てほしいのかが伝わりません。

詰まったときの一言

1. いま何をしようとしているか、3行で教えてください
2. 変更する前に、どのファイルをどう変えるかだけ先に見せてください（`Shift + Tab` でプランモード）
3. このエラーの原因を3つ挙げて、確からしい順に並べてください
4. さっきの変更を取り消して、元に戻してください（`Esc` 2回でも巻き戻せます）
5. 変更は app/page.tsx だけにしてください。ほかのファイルは触らないでください
6. 同じことを2回試して失敗しています。別のやり方を提案してください
7. ターミナルに出ているこの文章の意味を、専門用語なしで説明してください
8. この機能は今回は作りません。仕様書から外してください
9. 完成の条件を1つずつ実際に動かして確かめ、表で報告してください（`/check`）
10. いまの状態のまま止めてください。続きは私が決めます

やってはいけないこと

1. 実データを入れる。クライアント名、未公開の数字、個人情報。ここだけは例外なしです
2. 一度に5つ頼む。1回に1つ、直したら動かす。まとめて頼むと、どれが壊したのか分からなくなります
3. 「いい感じに」で頼む。判定できる言い方に直してから渡してください
4. 動かさずに完成とする。Veracode が150超のLLMに同一課題を解かせた調査では、コードの正しさが95%を超える一方、安全なコードが出た割合は約55%でした。画面が出たことは受け入れ基準になりません
5. ターミナルの `rm` でファイルを消す。チェックポイントは直近100個・30日を保持しますが、コマンドで消したファイルは戻りません

Claude Code 道具の作り方 チートシート

2026年8月21日 GenAI活用スキル向上支援 第3回／株式会社ギブリー 安田 光喜

どれを使うかの判断

Anthropic の公式ドキュメント（Extend Claude Code）が、足す順番を表で示しています。困りごとから引いてください。

困っていること	使う道具	置き場所
同じ規約を2回間違えた	CLAUDE.md	プロジェクト直下
同じプロンプトを打ち始めている	人が呼ぶ Skill	`.claude/skills/<名前>/SKILL.md`
同じ手順書を3回貼った	Skill に切り出す	同上
副次的な調べ物が会話を汚す	サブエージェント	`.claude/agents/<名前>.md`
毎回必ず起きてほしい	フック	`.claude/settings.json`
2つ目の案件でも同じ設定が要る	プラグイン	配布用リポジトリ

CLAUDE.md は200行以内が推奨です。長い規約は本体に書かず、CLAUDE.md には「この規約に従う」だけ残し、全文は Skill に置いてください。

コマンドとスキルは1つになりました

2026年の公式ドキュメントに「Custom commands have been merged into skills」と書かれています。`.claude/commands/deploy.md` と `.claude/skills/deploy/SKILL.md` は、どちらも `/deploy` を作ります。既存の `commands/` も動き続けますが、同じ名前なら Skill が勝ちます。付随ファイルを持てるのと、Claude が自分の判断で発火できるのは Skill 側だけです。

Skill の書式は agentskills.io としてオープン標準になりました。Cursor、GitHub Copilot、VS Code、OpenAI の Codex、Google の Gemini CLI など40社超が同じ `SKILL.md` を読みます。ツールを乗り換えても、書いた指示は持ち運べます。

SKILL.md の最小形

frontmatter に必要なのは `name` と `description` の2つだけです。

```
---
name: weekly-report
description: 週次報告のドラフトを作る。今週の実績メモを渡すと、決まった見出しで整える。「週次」「週報」
---

# 週次報告

## 手順
1. 実績メモを読む
2. 見出し（今週やったこと／数字／来週の論点）に振り分ける
3. 1項目1行で書く。形容詞は使わない
```

`name` は64文字まで。小文字の英数字とハイフンだけが使えます。`anthropic` と `claude` は予約語で使えません。`description` は1024文字まで。ここに発火の合図になる言葉を入れておくと、Claude が自分で拾います。

読み込みは3段階です。起動時に常時読まれるのは frontmatter だけで、1本あたり約100トークン。本文は発火したときだけ、付随ファイルは参照されたときだけ読まれます。道具を増やしても重くなりません。

サブエージェントの最小形

`.claude/agents/checker.md` に置きます。必須は Skill と同じく `name` と `description` の2つです。

```
---
name: checker
```

```
description: 仕様書の完成の条件を1つずつ確かめて報告する。修正はしない。
model: haiku
---

あなたは点検する係です。直す係ではありません。
`仕様書.md` の完成の条件を1項目ずつ実際に動かして確かめ、表で報告してください。
確かめられなかった項目は、正直にそう書いてください。推測で埋めないでください。
```

別の文脈で働き、要約だけが本体の会話に返ります。同時20体、1セッション200体まで。`model` を軽いものに落とすと費用が下がります。

フックのイベント名

31種類ありますが、実務で使うのはこの5つで足ります。

イベント名	いつ走るか	使い道
`SessionStart`	起動したとき	「実データを入れない」注意文を毎回出す
`UserPromptSubmit`	入力を送った瞬間	入力文に機密らしき文字列がないか検査する
`PreToolUse`	ファイル操作の直前	認証情報の書き込みを止める。ここだけが強制になる
`PostToolUse`	ファイル操作の直後	変更履歴を記録する。整形をかける
`SessionEnd`	終了したとき	作業ログを保存する

`type` は `command` / `http` / `mcp_tool` / `prompt` / `agent` の5種類。シェルスクリプトのほか、HTTP エンドポイントや LLM による判定も置けます。止めたいときはスクリプトを終了コード2で終わらせてください。

権限設定

`.claude/settings.json` に書きます。評価順は deny、ask、allow です。deny はどのスコープからも上書きできません。

```
{
  "permissions": {
    "deny": ["Read(//**/.env)", "Bash(rm:*)", "Bash(git push:*)"],
    "ask": ["Bash(curl:*)", "Write(//**)"],
```

```
    "allow": ["Read(./**)", "Edit(./app.html)"]
  }
}
```

覚えて帰る一点

CLAUDE.md や Skill に「.env を編集するな」と書いたものは、要望にすぎません。守られることもあれば、守られないこともあります。実際に止まるのは `PreToolUse` フックと `permissions.deny` だけです。プロンプトに書いた禁止は統制ではありません。統制にしたいなら、フックと権限設定に落としてください。

機密情報の扱い チートシート

2026年8月21日 GenAI活用スキル向上支援 第3回／株式会社ギブリー 安田 光喜

入れてよいもの、いけないもの

判断に迷う形で覚えると運用されません。文字列を見た瞬間に決まる形にしてください。

入れない	入れてよい
クライアント名、案件名、担当者名	業種と規模だけに落とした形（製造業X社、売上規模3,000億円想定）
未公開の財務数値、入札価格	公開済みの決算値、官公庁の統計
個人情報、要配慮個人情報	架空の氏名とダミーの連絡先
社内の未公開資料の本文	分析の型、指標の定義、集計ロジック
APIキー、パスワード、認証トークン	環境変数名だけ（値は書かない）

抽象化の型はひとつです。業種、規模、分析軸だけを残し、固有名詞と実数値を落とす。この形にできるなら渡せます。できないなら渡さないでください。

判断の物差しは3つです。この文字列だけを見てクライアントが特定できるか。この情報が5年間ベンダーのサーバーに残ってよいか。いまの会話がそのまま5年保存されるとして、問題がないか。

契約プラン別の学習利用と保持期間

分岐点は2025年8月28日の消費者向け規約改定です。同じ操作でも、契約が消費者向けか商用かで行き先が変わります。

区分	学習利用	保持期間
Free / Pro / Max（学習許可あり）	使われます	非識別化して最大5年
Free / Pro / Max（学習許可なし）	使われません	30日
Team / Enterprise / API / Bedrock 経由	既定で使われません	30日
安全性の審査でフラグが立った入出力	審査に使われます	最大2年
`/feedback` `/bug` `/share` で送ったもの	学習には使われません	5年

最後の行は見落とされがちです。プランに関係なく、`/feedback` は会話履歴とコードのコピーを送信し、5年保存されます。無効化する環境変数は `DISABLE_FEEDBACK_COMMAND=1` 。演習フォルダの設定には最初から入れてあります。

.gitignore は防御になりません

Claude Code は `.gitignore` も `.claudeignore` も読み取り制限として扱いません。The Register が2026年1月28日に検証しています。`.env` を書いた `.claudeignore` を置いた状態で v2.1.12 に読ませたところ、普通に読み、「このファイルには認証情報が含まれます」という警告を添えたうえで、内容をコンソールに出力しました。

効くのは `.claude/settings.json` の `permissions.deny` だけです。Read の deny は Edit も同時に塞ぎ、`cat` や `head` にも効きます。

permissions.deny の書き方

アンカーの違いで守れる範囲が変わります。ここを間違えると、書いたつもりで守れていません。

書き方	実際に守る範囲
`Read(.env)`	いま起動しているフォルダの直下だけ。親フォルダも別案件も対象外
`Read(//**/.env)`	ファイルシステム全体の `.env`

書き方	実際に守る範囲
`Read(./secrets/**)`	プロジェクト内の `secrets` フォルダ
`Read(/secrets/**)`（ユーザー設定に書いた場合）	`~/.claude/secrets/**` を指してしまいます

最小の書き方はこれです。

```
{
  "permissions": {
    "deny": [
      "Read(//**/.env)",
      "Read(//**/.env.*)",
      "Read(~/.aws/**)",
      "Read(~/.ssh/**)",
      "Read(//**/client-confidential/**)"
    ]
  }
}
```

Python や Node のスクリプトが自分でファイルを開く経路までは塞げません。そこまで止めるには、macOS の Seatbelt などOSレベルのサンドボックスが要ります。

手元のPCにも残ります

セッションの全会話は `~/.claude/projects/` に平文で30日残ります。暗号化されていません。クライアント名の入った会話をした場合、そのPCのディスクに平文で残るということです。保持日数は `cleanupPeriodDays` で変更でき、書き込み自体を止めるなら `CLAUDE_CODE_SKIP_PROMPT_HISTORY` を使います。

1案件1フォルダで起動する

運用ルールとして最も効くのがこれです。読み取りの射程は、起動したフォルダとその下に決まります。Desktop 直下や案件フォルダのルートで起動すると、隣のクライアントの資料まで射程に入ります。案件ごとにフォルダを分け、そのフォルダの中で `claude` を起動してください。

公開する前の3確認

1. 画面に出ている数字は、公開してよいものか

2. URL を知った人は誰でも開けます。認証を付けていないなら、それは公開です

3. 消したくなったときにどう消すかを、公開する前に確認したか