



Claude Code の mod でできること・作り方

公式ページ「Create a mod」ほかを、画面と図で読む操作説明書

出典：code.claude.com/docs/en/plugins/mods/（2026年10月2日に読んだ内容）。画面写真は公式ページの動画・画像から

原石工房

隠れた才能に気づき、花開く

ひとことで

Claude Code の画面と動きに、自分の機能を足せる

↓ ここを mod が足した (使ったツールの回数)

・ Thinking ・ **tool calls: 2...** (4s ・ ↓ 74 tokens)

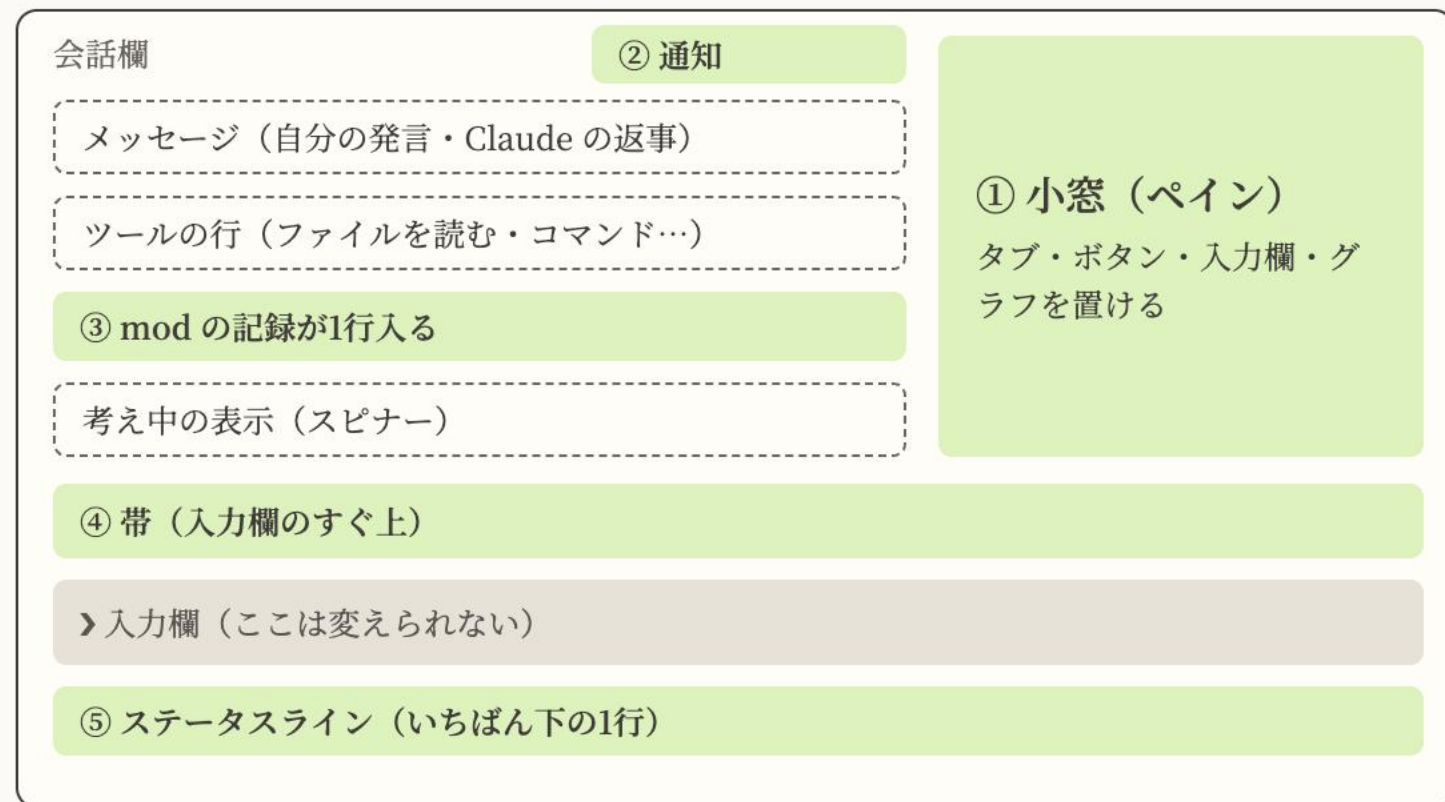
› █

▶▶ auto mode on (shift+tab to cycle) ・ esc to interrupt ・ ← for agents

Claude が考えている間の表示に、ツールを使った回数を足した例。本体には手を入れず、数行のプログラムを足しただけ。

画面のどこを変えられるか

緑の場所に足せて、点線の場所は描き直せる



新しく足せる場所

元からある部分を描き直せる

変えられない

許可を聞く確認画面も変えられない。
画面が狭いと、小窓は入力欄の上に出る。

できること

公式が挙げている5つ

- | | | |
|---|-------------|-------------------------------------|
| 1 | 画面を足す | 小窓・帯・ボタン・入力欄。例：返事のたびに会話の容量をグラフで出す |
| 2 | 元の画面を描き直す | ツールの行・考え中の表示・Claude が質問してくる画面 |
| 3 | ツールの実行に割り込む | 止めて自分に聞く／実行せずに代わりに答える／別のモデルに送る |
| 4 | 自分のコマンドを足す | / で始まるコマンド。Claude の返事を待たず、作業中でもすぐ動く |
| 5 | 数えて、見せる | 1つのフックが記録したものを、別のフックが表示できる |

実例1 | 小窓

コマンド1つで、ボタン付きの小窓を開く

1 /hello-tabs と打つ

```
/hello-tabs          Open the hello-tabs pane
> /hello-tabs
>> auto mode on (shift+tab to cycle)
```

2 タブ付きの小窓が開く

```
1: One  2: Two      ×
This is the first tab.

> 
>> auto mode on (shift+tab to cycle) · ← for agents
```

3 2 で切り替え、a で数が増える

```
1: One  2: Two      ×
[ Add one ] Count: 2

> 
>> auto mode on (shift+tab to cycle) · ← for agents
```

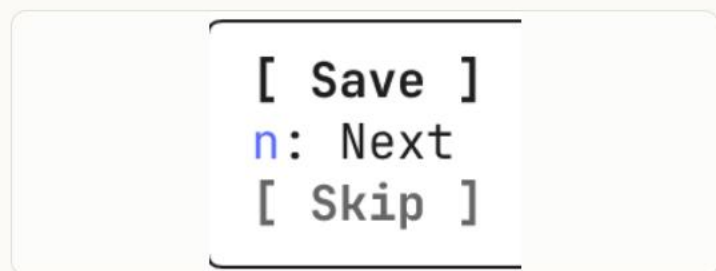
数は保存されるので、Claude Code を閉じて開き直しても残っている。

Esc で小窓を閉じる。

実例2 | 使える部品

小窓や帯に置ける部品（実際の画面）

ボタン



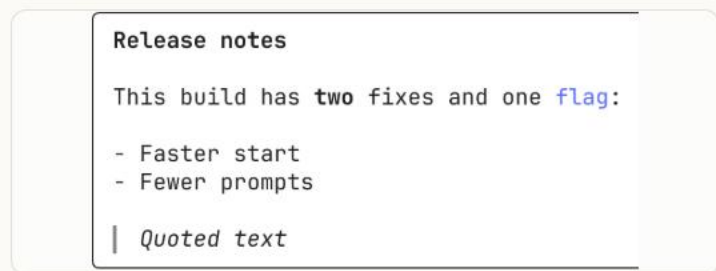
文字の入力欄



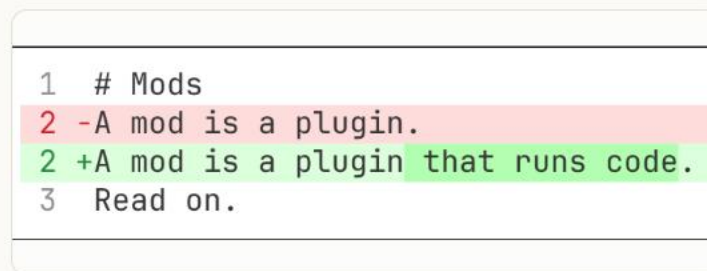
選択リスト



書式付きの文章



変更の差分



色のマス



Claude Code の機能そのものも mod で作られている

/diff 組み込み

変更点を小窓で見せる。ボタンをキーに割り当て、スクロールも mod 自身が受け持つ

you-should-know 組み込み・初めは無効

長い作業の横で別のエージェントが見張り、見落としそうなことを入力欄の上に知らせる

token-weather 有志が公開

会話の残り容量の「予報」を帯に出す

blast-radius 有志が公開

危ないコマンドを実行する前に試し打ちして、止めるかを確かめる

有志の2つは GitHub の説明文を読んだだけで、中身は確かめていない。

思いつく使い方（案）

この環境なら、たとえばこう使える

1 朝のまとめを帯に

毎朝の Slack まとめの要点1行を、入力欄の上に出しておく

2 消す前に止める

削除のコマンドが走る前に止めて、「実行する／やめる」のボタンを出す

3 今日の残りを小窓に

X の投稿枠や API の使用量を、色のマスでグラフにする

4 すぐ動く自分のコマンド

/予定 と打つと、Claude を待たずに今日の予定を表示する

どれもまだ作っていない。できるかどうかは、作る前に確かめる。

他の仕組みとの違い

画面に描けるのは mod だけ

	mod	設定のフック	スキル	MCP
中身	JavaScript・TypeScript	シェルのスクリプト	手順を書いた文章	外部のサーバー
画面に描ける	できる	できない	できない	できない
変えられるもの	ツール・入力・コマンド・画面	ツールや入力を通すか止めるか	Claude が知っていること	Claude が使える道具
向いている場面	小窓・帯・自分のコマンドが欲しい	手持ちのスクリプトで止めたい・記録したい	同じ指示を何度も貼っている	外のサービスにつなぎたい

1つのプラグインに4つとも入れられる。スキルで足りるなら、まずスキルで。

作り方A | Claude に頼む

書いてもらって、許可して、試すだけ

入力欄にこう書く

› 入力欄の上に、今の git ブランチ名を出す mod を作って

1

頼む

作りたいものを自分の言葉で書く

2

許可する

ファイルごとと、読み込むときに聞かれる

3

確かめる

/plugin の Installed タブに並ぶ

4

試す

違ったら直す点を伝える。すぐ反映

Claude Code v2.1.287 以降が必要 (`claude --version` で確かめる)。Claude は組み込みのスキル `plugin-authoring` を見ながら書く。

最初の保存のとき、どちらかを選ぶ

Enable for this session

このセッションで読み込む

- Claude の返事が終わったら読み込む
- 直すたびに読み込み直す
- セッションを再開しても有効

Not now

今は読み込まない

- ファイルは残る
- 次にそのセッションを開いたとき読み込む
- 使わないならフォルダごと消す

その前に、ファイルを1つ作るごとに許可を聞かれる（`~/.claude` は守られたフォルダのため）。

作り方A | 作ったあと

作った mod は、そのセッションでしか動かない

作った場所

~/.claude/dev-mods/
セッションID/git-branch/
古くなると、フォルダごと消える

コピー
➡

自分の場所

~/mods/git-branch/

起動
➡

どのセッションでも

```
claude --plugin-dir  
~/mods/git-branch
```

人に配るとき

マーケットプレイスに載せ、[/plugin](#) で入れてもらう。名前を [claude-](#) で始めると落とされる。試した Claude Code の版を README に書いておく。

作り方B | 自分で書く

書くファイルは3つだけ

例題：使ったツールの回数を数え、考え中の表示に出し、`/tally` と打つと答える mod

```
first-mod/  
├── .claude-plugin/  
│   └── plugin.json ← ①名札  
└── hooks/  
    ├── hooks.json ← ②案内  
    └── register.js ← ③中身
```

① plugin.json

名前・版・説明を書く名札

② hooks.json

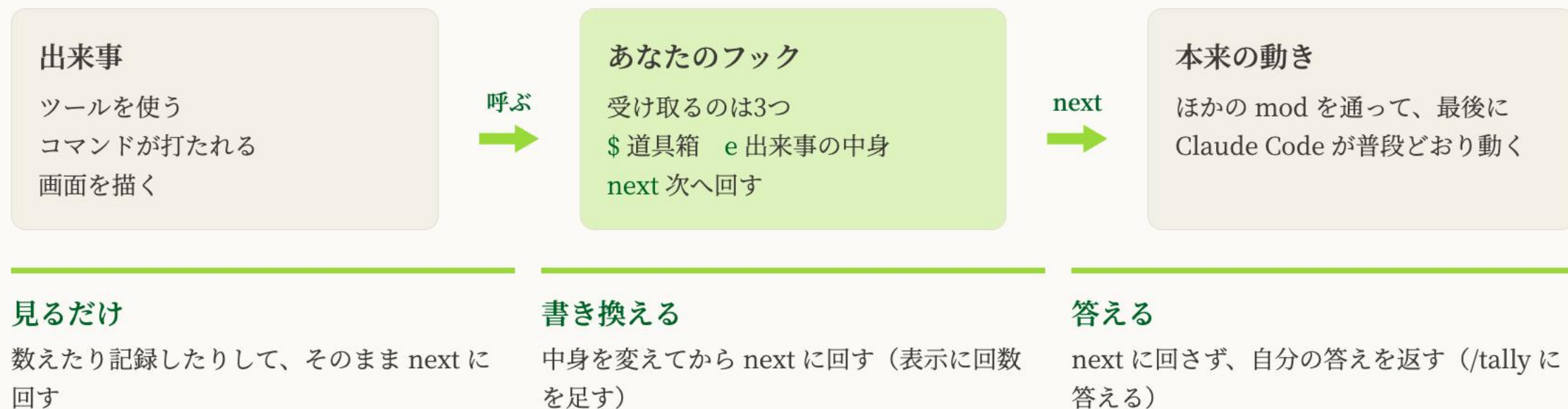
「中身は register.js」と指す。これがあると mod になる

③ register.js

どの出来事で、何をするかを書く

Node.js もビルドも要らない。js や .ts をそのまま読み込む。

出来事が起きると、Claude Code があなたの関数を呼ぶ



それぞれが画面のここに効く

```
• Thinking • tool calls: 2... (4s • ↓ 74 tokens)
```

```
› █
```

```
›› auto mode on (shift+tab to cycle) • esc to interrupt • ← for agents
```

```
› list the files here and read the README
```

```
Read 2 files, listed 1 directory
```

- The project holds `README.md`, `package.json`, and the `first-mod` directory. The README describes a small Express service with one health check.

```
※ Worked for 8s • done 8:46 PM
```

```
› /tally
```

```
L first-mod: Claude has made 3 tool calls since this mod loaded
```

- ① 始まったとき
/tally を登録する
- ② ツールを使う直前
回数を1増やす
- ③ /tally と打ったとき
回数を文で答える（下の画面）
- ④ 考え中の表示を描くとき
回数を足す（上の画面）

作って、読み込んで、試して、直す

1 フォルダを作り、3つのファイルを保存

```
mkdir -p first-mod/.claude-plugin first-mod/hooks
```

2 インストールせずに読み込む

```
claude --plugin-dir ./first-mod
```

3 何か頼んでから、/tally と打つ

```
first-mod: Claude has made 2 tool calls since this mod loaded
```

4 開いたままコードを直して保存

```
tool calls → tools used に書き換えると、すぐ表示が変わる
```

読み込み直すと回数は0に戻る。/tally が一覧に出なければ、読み込めていない。

register.js は、この形をしている

```
let calls = 0

export function register(on) {
  on('session.start', ...)           // ① /tally を登録
  on('tool.call', ...)               // ② calls を1増やす
  on('command.run', { command: 'tally' }, ...) // ③ 回数を答える
  on('ui.render', { component: 'Spinner' }, ...) // ④ 表示に足す
}
```

`on(出来事の名前, 関数)` を並べるだけ。2つめの `{ command: 'tally' }` は「このときだけ動く」という絞り込み。

全文はノート欄と公式ページの手順4にある。

作ったあと

入れる前に、中身を一覧で確かめられる

```
$ claude plugin validate ./first-mod

hooks: session.start, tool.call,
       command.run{command=tally}, ...
calls: $.command.register, $.ui.invalidate
```

✓ Validation passed

hooks: 使う出来事 **calls:** 呼ぶ道具
他人の mod も、入れる前にこれで見る。

試験する

claude plugin test

通信もログインもなしで動く

一覧表を見る

読み込むたびに **.claude-plugin/types/** に書かれる。
説明ページと食い違ったら、こちらが正しい

使うときの注意

この環境で気をつける3つ

見える場所

小窓や帯が見えるのは、Mac のターミナルとデスクトップアプリの Code タブ。Windows・iPad から Remote Control で操作しても、表示は Mac 側の画面に出る

権限

mod は自分と同じ権限で動き、ファイルも鍵も読める。他人の mod は、入れる前に validate で中身を見る

止め方

1つ → `/plugin` で無効。そのセッションだけ → `--safe-mode`。 `disableAllHooks` は全部止まるが、削除を止めるフックやステータスラインも一緒に止まる



次に読むページ

code.claude.com/docs/en/plugins/mods/ の下

Draw in the interface

小窓・帯・ボタン・入力欄

Use the mods API

コマンド・モデル呼び出し・タイマー

Troubleshoot a mod

動かないときの調べ方

React to events

ツール・入力・返事の出来事

Interface gallery

部品の見本と画面写真

組み込み mod のソース

GitHub anthropics/claude-code の mods

最初の1本、画面のどこに何を出しますか。

