

# AIと共に存する時代の学習法

Claude Codeの学習モードを例に

# 自己紹介

- azukiazusa
- <https://azukiazusa.dev>
- FE (フロントエンド|ファイアーエムブレム) が好き



## 本日の問いかけ

「AIに聞けば3秒で答えが出る時代に、なぜ私たちは学び続ける必要があるのか？」

# 今日話す内容

- AIの登場により懸念されている「思考力の低下」について
- AIとどのように付き合い、学び続けるべきか
- 学習モードはどのように機能し、どのような効果があるのか
- 学習モードの具体例としてClaude Codeの学習モードを紹介

# 知識の獲得方法の変化

## Before ChatGPT

- Google検索
- 周囲の人質問
- 図書館に向いて関連書籍を調査

情報を得るために自ら能動的に動き、取捨選択を行う必要があった

## After ChatGPT

- 自然言語で質問すると、瞬時に答えが得られる
- 情報収集の手間が大幅に削減された

受動的に答えが得られる

# コードの大部分はAIが生成するようになった

- Claude Code や Devin といった AI コーディングエージェントの登場
- 開発者は自然言語で指示を出すだけでエージェントが自律的にコードを生成してくれる、いわゆる「Agentic Coding」の時代が到来
- これにより、コードの大部分は AI が生成するようになり、開発者はコードの生成からコードのレビューや設計、AI エージェントの管理へと仕事の内容がシフトしつつある
- AIが自動生成したコードをベースに、開発者が微調整を加えるスタイルが主流に

# 従来の問題解決例

問題発生： React useEffect を使ったら無限ループが発生した



Google検索「React useEffect infinite loop」



Stack Overflow の回答や技術ブログを複数参照



公式ドキュメントを熟読



試行錯誤（3-4回のエラーエラー体験）



解決（所要時間：2-4時間）



副産物：

- なぜ依存関係の警告が出るのか理解できた
- useEffectの挙動についての理解が深まった

# 問題の本質を見失うリスク

問題発生



「React useEffect を使ったら無限ループが発生したので、修正して」



解決策が瞬時にコードに反映される（所要時間：数秒）



副産物：なし？依存への慣れ？

短期的な解決は得られるが、長期的な学びの機会を失う可能性

# コード生成の便利さと落とし穴

## 便利さの実例

```
// AIに「React TODOアプリを作って」と依頼
// → 5分で完成度の高いアプリが生成

function TodoApp() {
  const [todos, setTodos] = useState([]);
  const addTodo = (todo) => {
    setTodos([...todos, todo]);
  };
  // 実際に動作するコードが一瞬で...
}
```

# コードの意図を理解できていない

// 理解度の実態

質問：「useState([])の[]は何を意味しますか？」

回答：「えーっと…初期値？」

質問：「setTodosを呼ぶとき、.push() メソッドを使わないのはなぜ？」

回答：「よくわかりません…」

質問：「このコードのパフォーマンス上の問題点は？」

回答：「うーん…」

問題：動作するコードは書けるが、なぜ動くかが分からぬ

## なぜ本質的な理解が重要なのか？

🤔 AIエージェントがコードを書くようになれば、自身の力で解決できなくても問題ないのでは

- コードレビューが仕事の大部分を占めるようになる
  - AIが生成したコードの品質を評価し、改善点を指摘する能力
- 最終的な責任は人間が負う
  - コードを保守していくためには、コードがどのように動いているのかを理解している必要がある
- 似たような問題に直面した際に、適切な解決策を自分で考え出す能力

# 「グライダー人間」から「飛行機人間」へ

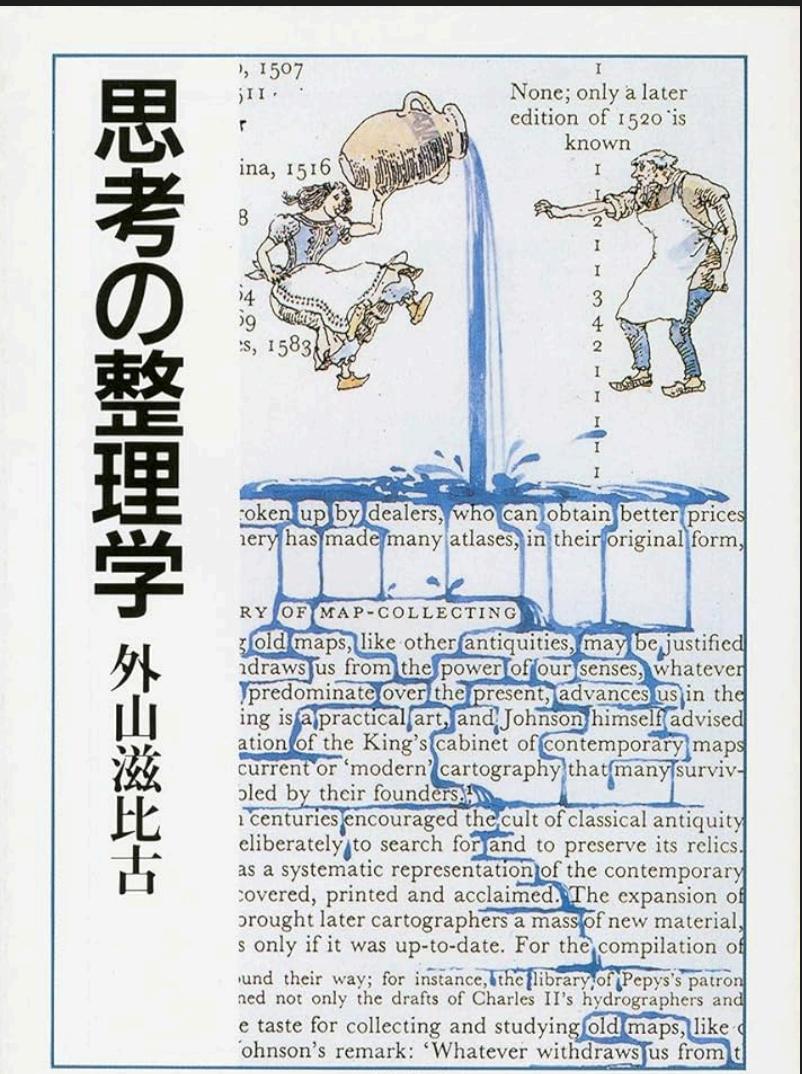
# グライダー人間（受動的な学習者）

- 他者に引っ張られて飛ぶだけ
  - AIに答えを聞くだけの使い方では、いずれAIに取って代わられる可能性

# 飛行機人間（能動的な思考者）

- 自力で飛翔できる
  - 自ら考え、創造する力を持つ
  - AIを使って+αの価値を生み出す

<https://www.chikumashobo.co.jp/product/9784480020475/>



# VUCA時代に求められる能力

VUCA = Volatility (変動性) , Uncertainty (不確実性) , Complexity (複雑性) , Ambiguity (曖昧性)

- 問題を解決する能力だけでなく、問題を発見する能力が重要に
- AIは既知の問題を効率的に解決することに長けているが、未知の課題を見つけ出したり、創造的な価値を生み出すことは人間の仕事

# 生成AIの盲信

AIが生成したコードを無批判に受け入れてしまう危険性

- セキュリティ上のリスク
- 技術的負債の蓄積

批判的思考力を身につけて、AIを適切に活用しよう

# 劇場のイドラー (Theater of Idols)

フランシス・ベーコンの「4つのイドラー」より

権威や流行に盲従してしまう認知バイアス

AI時代における「劇場のイドラー」

- **AIの権威性**：「AIが言うから正しい」
- **技術の流行性**：「最新のAIツールなら間違いない」
- **自動化への過信**：「機械の方が人間より正確」



権威への盲従は  
判断力を奪う

# 批判的思考力の重要性

批判的思考：与えられた情報や状況を鵜呑みにせず、多角的な視点から分析し、論理的に考察して、客観的な判断や評価を行うための思考力

## AI時代に必要な批判的思考

- **出力の検証**：生成されたコードの妥当性確認
- **複数の視点**：異なるアプローチの検討
- **根拠の確認**：なぜその解決策なのかを自分の言葉で説明できるように

<https://www.keisoshobo.co.jp/book/b534036.html>

## 思考力改善ドリル

批判的  
思考 から  
科学的  
思考へ



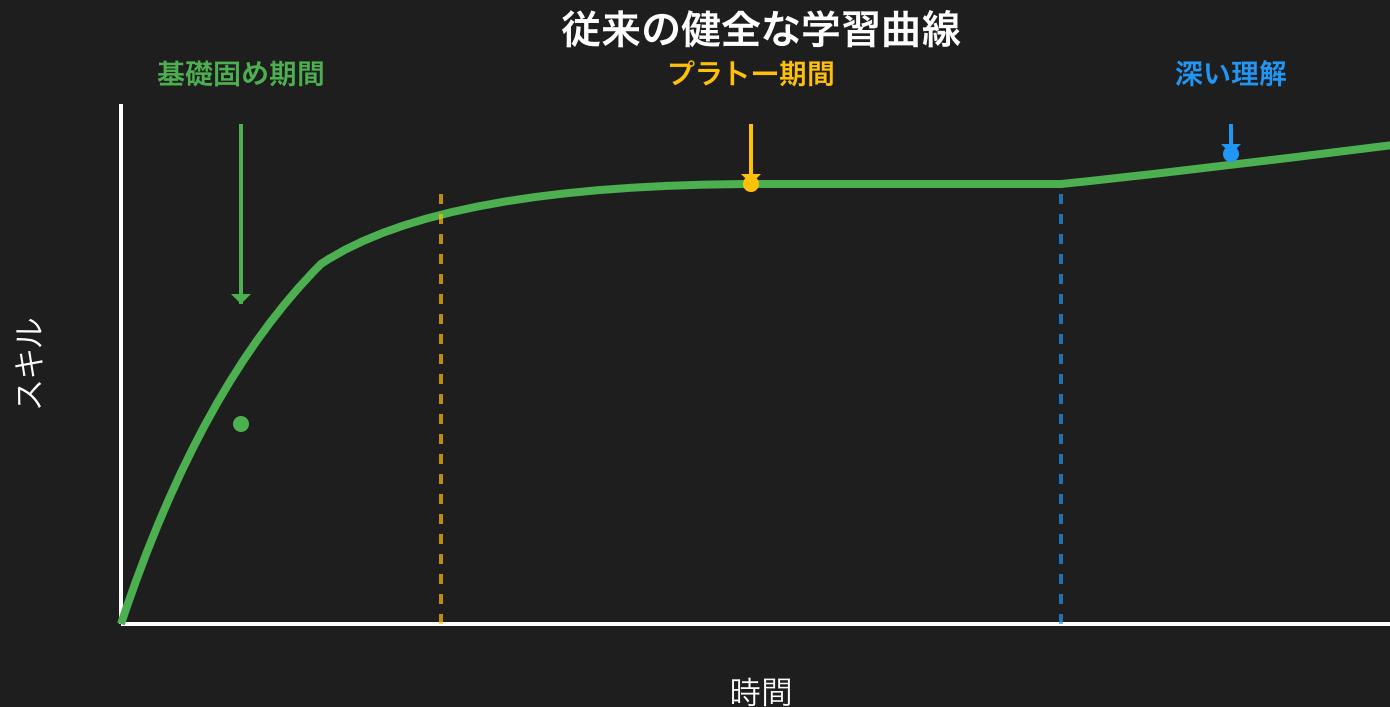
植原亮

クイズ感覚で問題を解いて  
クリティカル・シンキングの力を養い、  
科学リテラシーがぐんぐん身につく！  
考える力を磨くための27章。



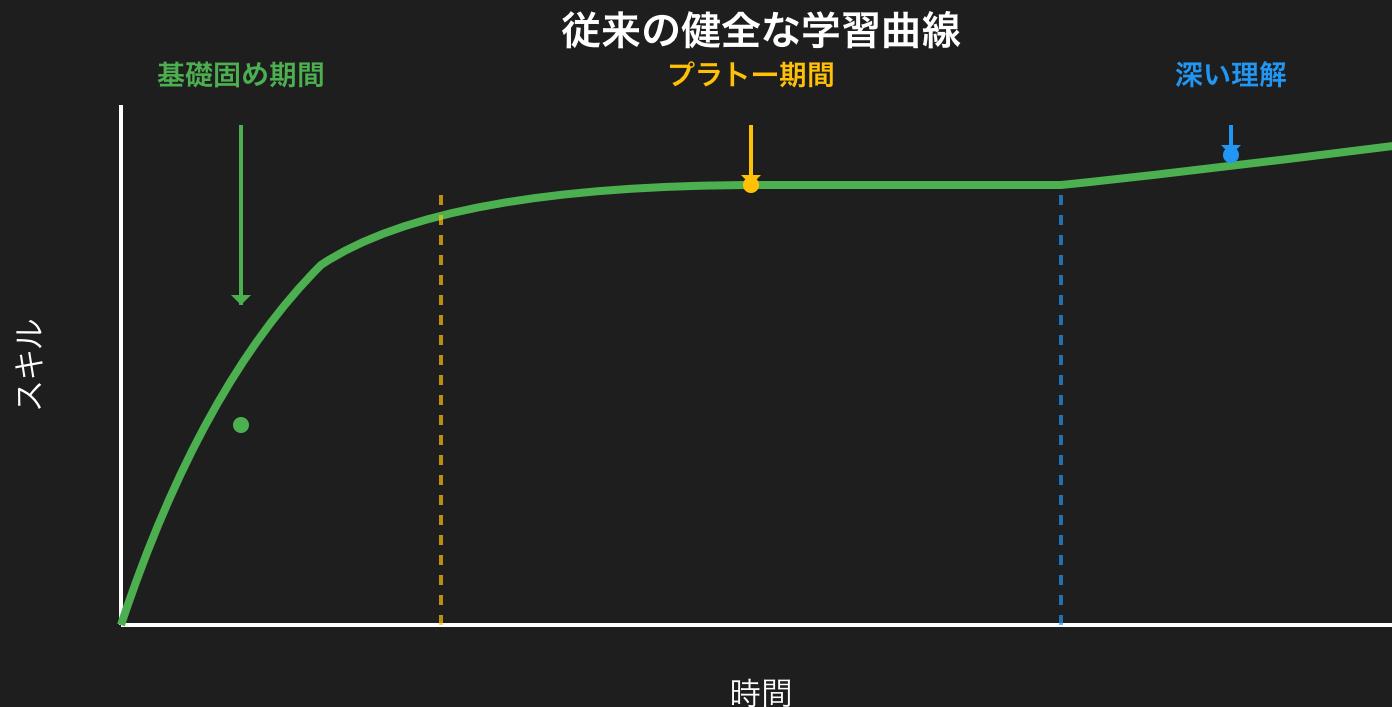
# 学習曲線の変化

## 従来の健全な成長パターン



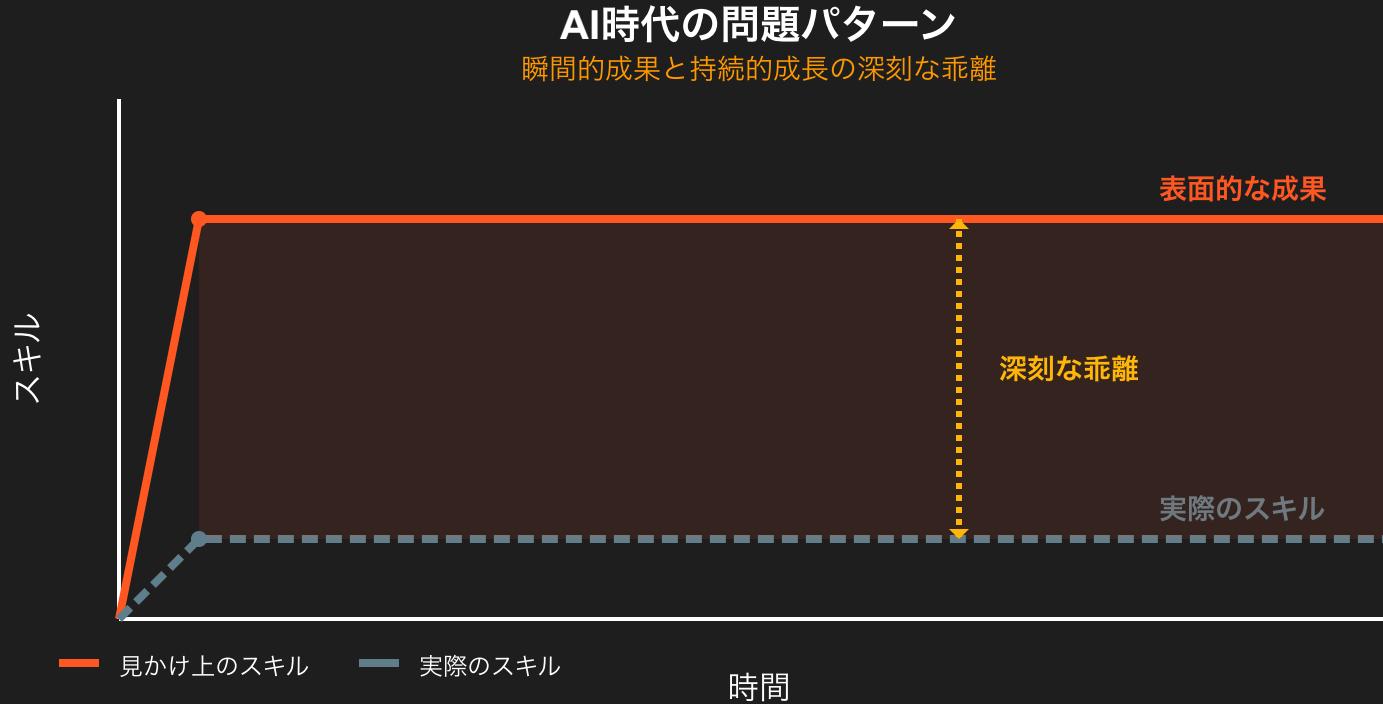
# 学習曲線の変化

## 従来の健全な成長パターン



- プラトーペリオドは、脳が知識を整理・成熟させる期間
- この期間を乗り越えることで、さらなる応用力や思考速度の向上につながるとも言われている

# AI時代の問題パターン



基礎が伴わないまま表面的な成果だけが得られる

## AIとの理想的な関係性

「答えを出す道具」から「思考を深めるパートナー」へ

→ 元からよく思考してた人は更に深く考えるようになり、AI時代により差がつく

## 目指すべきAI活用像

- AIを「答えを得るため」ではなく、「思考を深める（=学ぶ）」ために使う
- よりよい出力を得るために問い合わせを立てる
- AIの解答を鵜呑みにしない批判的思考

AIをパートナーとして活用するために、AIを学習に使う

# 各社の学習支援機能

## ChatGPTの学習モード

すぐに解答を示すのではなく、ソクラテス式問答法（対話を通じて自ら考えさせる方法）やヒント、問いかけを組み合わせ、学生自身の考える力を導きます。これにより、理解を深め、能動的な学習を促進します。

## Geminiのガイド付き学習

ガイド付き学習は、掘り下げた質問や自由回答形式の質問を通して参加を促し、議論を促し、テーマをより深く掘り下げる機会を提供します。その目的は、単に答えを得るだけでなく、深い理解を育むことです。

## Claude for Education

直接答えるのではなく発見を導く、ソクラテス式質問を通して思考力を養う、解決策ではなく原則に焦点を当てる

# AIをメンターとして活用する

- AIからの質問に答える形式で自分の頭で考える力を養う
  - 曖昧に理解した知識が明確になる
  - 答えを出す過程が重要
- AIをメンターとして活用することの利点
  - 恥を欠かない→素直に質問できる
  - 24時間365日、低コストでいつでも利用できる

# AIメンターの利点 1：恥をかかない環境での学習

## 従来の学習環境での課題

- 質問への恐れや不安
  - 間違った内容の質問をしていいなか
  - 他の人にどう思われるか
- 自尊心が邪魔をする
  - 自分をよく見せたいという気持ち
  - わからないことを認めたくない
- 結果として疑問をそのままにしてしまう

## AIメンターの利点

- 周囲の目を気にしない
- 素直に質問できる
- 自分の考えていることの言語化に集中できる

## AIメンターの利点2：24時間365日、いつでも利用可能

### 従来の学習支援の制約

- 時間的制約
  - 先生や講師の都合に合わせる必要
  - オフィスアワーや営業時間内のみ
- コスト的制約
  - 家庭教師は高額な費用
  - 個別指導は限られた人のみ
- 地理的制約
  - 居住地域による学習機会の格差

### AIメンターの利点

- いつでも質問できる環境
- 自分のペースで学習を進められる
- 低コストで利用可能
- **学習機会の民主化**

# 実際の学習モードの例

コーヒーと目覚め ▾

A なぜコーヒーを飲むと目が覚めるの

コーヒーが目を覚ますメカニズムについて学びましょう！まず、あなたの現在の理解を確認したいと思います。

コーヒーに含まれる成分で、目覚ましの効果があると思われるものを知っていますか？また、その成分が体の中でどのような働きをするとと思いますか？

さらに、なぜ人は普通に眠くなるのか、その仕組みについて何かご存知でしょうか？

これらの質問を通して、コーヒーの目覚まし効果の科学的なメカニズムと一緒に探っていきましょう。

A カフェインは知っているけれど、どのように作用するかは知らない

完璧です！カフェインが重要な成分であることはご存知なんですね。では、このメカニズムと一緒に探っていきましょう。

まず基本から考えてみましょう。人が眠くなる時、体の中では何が起こっていると思いますか？つまり、「眠気」そのものはどのような仕組みで発生すると想像されますか？

そして、カフェインがその「眠気のメカニズム」にどう干渉する可能性があるかについて、何かアイデアはありますか？

Reply to Claude...

Claude Sonnet 4 ▾

コーヒーと目覚め ▾

完璧です！カフェインが重要な成分であることはご存知なんですね。では、このメカニズムと一緒に探っていきましょう。

まず基本から考えてみましょう。人が眠くなる時、体の中では何が起こっていると思いますか？つまり、「眠気」そのものはどのような仕組みで発生すると想像されますか？

そして、カフェインがその「眠気のメカニズム」にどう干渉する可能性があるかについて、何かアイデアはありますか？

Researching...  
Interrupted · 611h 36m

A 人が休憩を必要とする場合に眠くなる？

興味深い仮説ですね！休憩と眠気の関係について考えてくださったのは素晴らしいアプローチです。

確かに体が休憩を必要とする時に眠くなることはありますが、もう少し深く考えてみましょう。例えば：

- 朝起きたばかりなのに、昼食後に眠くなることはありませんか？
- 逆に、とても疲れているのに夜眠れないことありませんか？

これらの現象を考えると、「休憩が必要だから眠くなる」以外にも何か別のメカニズムがありそ

Reply to Claude...

Claude Sonnet 4 ▾

# なぜ一人で学習するのではなく、AIとともに学習するのか？

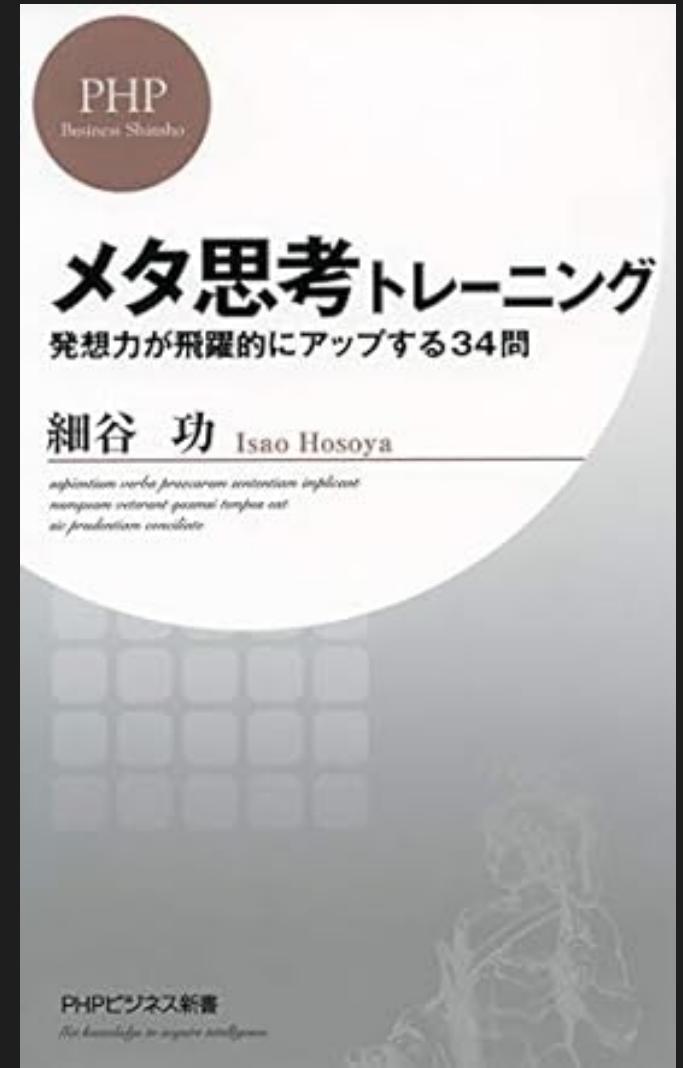
- 多様な視点の提供
  - 自分では気づかない観点を教えてくれる
  - 異なるアプローチを提案してくれる
- 学習の道筋を示してくれる
  - 適切な質問で思考を導いてくれる
  - 段階的な理解を促進してくれる
- 対話による理解の深化
  - 説明することで自分の理解度を確認
  - 質問を通じて知識の穴を発見

# メタ認知の重要性

## 自分の思考プロセスを客観視する能力

- 自分の理解度を把握する → 不足している知識を特定する
- 自分の頭で考えることで、どの部分が理解できていないかを認識する
- AIが自分では気づかない視点を提供してくれる

<https://www.php.co.jp/books/detail.php?isbn=978-4-569-82773-5>



## コーディングエージェント（Claude Code）の学習モード

ユーザーが自分の力で考えてコードを書く練習をすることを重視

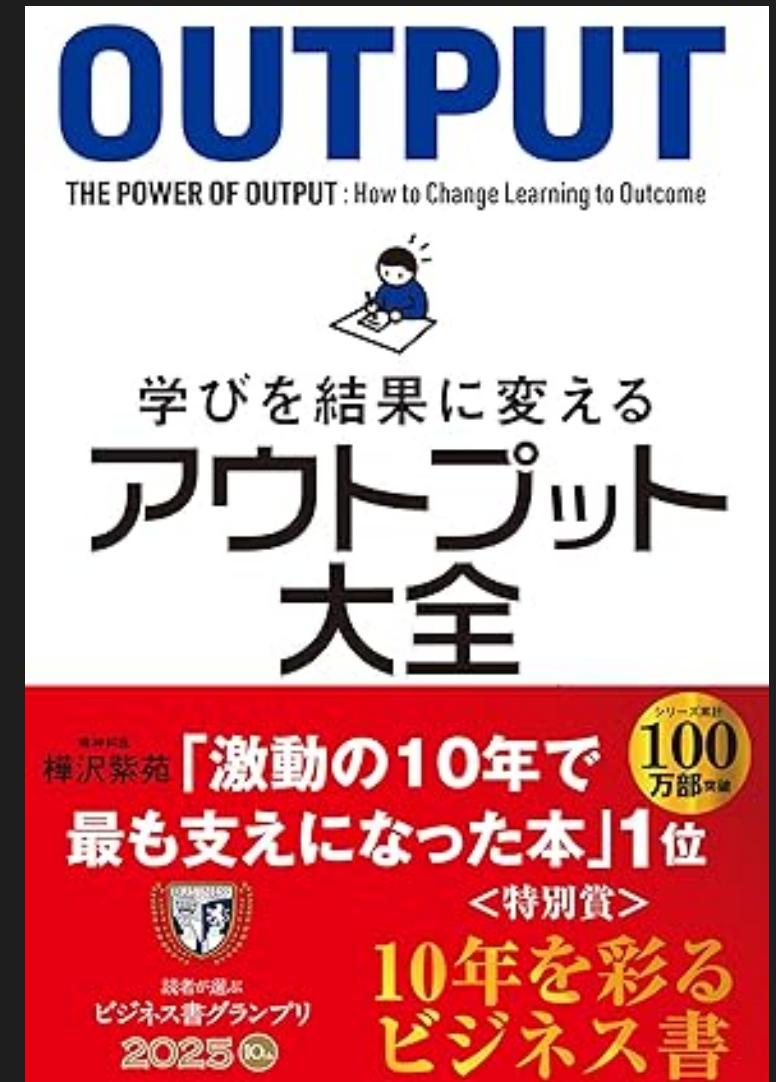
- AIがなぜそのコードを書いたのかを「Insight」として説明
- AIがすべてのコードを生成するのではなく、ユーザーにヒントを与え、コードを自分で書かせる
- どのようなコードを書くべきか思考する余地を残す

試行錯誤を経験し、問題解決能力を養う経験はAIに頼り切ってしまった得られない学び

# 自分で手を動かした経験は長期的に記憶に残る

- ・インプット（読む、聞く）だけでは記憶に定着しない
- ・アウトプット（書く、話す、行動する）を通じて初めて定着する
- ・誰かに説明するという過程で自分の曖昧な理解が明確になり、知識がより深く定着する
- ・コードをそのまま書き写す「写経」と呼ばれる学習法も長く知られてきた

<https://www.sanctuarybooks.jp/book/detail/1018>



## 長期的な投資としての学び

- 「タイパ」という観点では、AIにコードを書かせる方が速い
- しかし、AIに頼り切ってしまうとコードを書く経験が積めず、長期的にはスキルが伸び悩む可能性
- 短期的に成果が出る速度は落ちるかもしれないが、将来の自分への投資と考える

# Claude Code学習モード デモ

課題：Next.js でカンバンアプリケーションに機能を追加する

## output-style を設定

- Claude Code の学習モードは `output-style` の 1 つ
- `output-style` は Claude Code のシステムスタイルを直接変更する
- デフォルトで以下の3つのスタイルが用意されている
  - `default`
  - `explanatory` (説明型) : Claude が選択したコードの実装方針やコードパターンについて説明しながらコードを生成する
  - `learning` (学習型) : Claude がユーザーの学習をサポートするために、「インサイト」を共有したり、一部のコードの生成をユーザーに依頼する



# Learning を選択

```
* Welcome to Claude Code!
```

```
/help for help, /status for your current setup
```

```
cwd: /Users/asai/sandbox/kanban-app
```

## Choose your preferred output style:

This changes how Claude Code communicates with you

Use /output-style:new to create custom output styles

- 1. Default Claude completes coding tasks efficiently and provides concise responses✓
- 2. Explanatory Claude explains its implementation choices and codebase patterns
- 3. Learning Claude pauses and asks you to write small pieces of code for hands-on practice

Enter to confirm · Esc to exit

## プロンプト例

カンバンボードにおいて、表示中のタスクを文字列でフィルタリングする機能を実装してください。

- ユーザーが入力フォームに入力した文字列に基づいて、表示中のタスクをリアルタイムでフィルタリングする
- タスクのタイトルにフィルタ文字列が含まれている場合にそのタスクを表示する
- 入力フォームはカンバンボードの上部に配置する

# 実行計画を立てる

- ユーザーがハノノオームにした文字列に基づいて、表示中のタスクをリアルタイムでフィルタリングする
  - タスクのタイトルにフィルタ文字列が含まれている場合にそのタスクを表示する
  - 入力フォームはカンバンボードの上部に配置する
- カンバンボードにタスクフィルタリング機能を実装しますね。まず、現在のコードベースを理解するために、プロジェクト構造を確認して実装計画を立てましょう。
- **Update Todos**
    - └ □ 現在のカンバンボード 実装を調査する
      - タスクフィルタリングのためのstate管理を設計する
      - フィルタ入力フォームコンポーネントを実装する
      - タスクフィルタリングロジックを実装する
      - カンバンボード表示にフィルタ機能を統合する
  - **List(app)**
    - └ Listed 13 paths (ctrl+r to expand)
  - **Read(app/boards/[boardId]/page.tsx)**
    - └ Read 60 lines (ctrl+r to expand)
  - **Read(components/drag-drop-board.tsx)**
    - └ Read 241 lines (ctrl+r to expand)
  - \* **Zesting...** (esc to interrupt)
    - └ Tip: Use /statusline to set up a custom status line that will display beneath the input box

# インサイトが表示される

- ★ Insight ●  
現在のカンバンボード実装では、データはサーバーから取得し、クライアント側でuseOptimisticを使って楽観的更新を実装しています。フィルタ機能は純粋にクライアント側の表示ロジックなので、React stateでフィルタ条件を管理し、表示時に動的にフィルタリングするのが効率的です。
- Update Todos
  - └ ✎ 現在のカンバンボード実装を調査する
  - ✎ タスクフィルタリングのためのstate管理を設計する
  - フィルタ入力フォームコンポーネントを実装する
  - タスクフィルタリングロジックを実装する
  - カンバンボード表示にフィルタ機能を統合する
- Zesting... (esc to interrupt)
  - └ Tip: Use /statusline to set up a custom status line that will display beneath the input box

> █

? for shortcuts



# ユーザーの入力を促す

The screenshot shows a code editor with several tabs: schema.prisma, board.ts, drag-drop-board.tsx, and Claude Code. The drag-drop-board.tsx tab is active, displaying the following TypeScript code:

```
components > drag-drop-board.tsx > DragDropBoard
32  export function DragDropBoard({ board: initialBoard }: DragDropBoardPr...
262  <div className="relative max-w-md">
263    <Search className="absolute left-3 top-1/2 transform -translate-y-1/2" placeholder="タスクをフィルタ..." value={filterText} onChange={(e) => setFilterText(e.target.value)} className="pl-10" />
264  </div>
265
266  /* TODO(human) - Add filter result information here */
267
268  <DndContext sensors={sensors}>
269    <DragOverlays onDragStart={handleDragStart} onDragOver={handleDragOver} onDragEnd={handleDragEnd}>
270      <div className="flex gap-6 overflow-x-auto pb-4">
271        {filteredBoard.columns.map((column) => (
272          <DraggableColumn column={column} />
273        ))
274      )
275    </div>
276  </DndContext>
277
278  <DragOverlay>
279    {draggedTask ? (
280      <div className="rotate-3 scale-105">
281        <DraggableTask task={draggedTask} />
282      </div>
283    ) : null}
284
285
286
287
288
289
290
291
292
293
```

The right side of the interface is a 'Claude Code' panel with the following content:

- Read 15 lines (ctrl+r to expand)
- Update(components/drag-drop-board.tsx)
  - Updated components/drag-drop-board.tsx with 2 additions
    - >
    - </div>
    - + /\* TODO(human) - Add filter result information here \*/
    - + <DndContext sensors={sensors} onDragStart={handleDragStart} onDragOver={handleDragOver} onDragEnd={handleDragEnd}>
    - + <div className="flex gap-6 overflow-x-auto pb-4">
    - + {filteredBoard.columns.map((column) => (
    - + <DraggableColumn column={column} />
    - + </div>
- Update Todos
  - Analyze current kanban board structure and components
  - Add filter input form to the top of kanban board
  - Implement client-side filtering logic for tasks
  - Integrate filter functionality with existing task display
  - Fix drag-and-drop functionality with filtered data
  - Request human input on filter result display implementation
- Learn by Doing
  - Context: フィルタリング機能の基本的な実装は完了しましたが、ユーザビリティを向上させるために、フィルタリングの状態を示す情報を追加したいと思います。現在のフィルタ結果で表示されているタスクの総数や、フィルタがアクティブな状態であることを明確に示すことで、ユーザーがより効果的にフィルタ機能を活用できます。
  - Your Task: /Users [REDACTED]/sandbox/kanban-app/components/drag-drop-board.tsx のフィルター入力フィールドの下に、フィルタ結果の表示情報を追加してください。TODO(human)コメントの場所に実装してください。
  - Guidance: フィルタされたタスクの総数を表示し、フィルタがアクティブ (filterTextが空でない) な場合は「〇件のタスクが見つかりました」のようなメッセージを表示してください。filteredBoardから各columnのtasks.lengthの合計を計算することができます。フィルタが無効な場合は何も表示しないか、全タスク数を表示するかを検討してください。

# フィードバック

```
57 267      onChange={(e) => setFilterText(e.target.value)}  
58 268      className="pl-10"  
59 269    />  
60 270  </div>  
61 271  
62 272  {filterText && (  
63 273    <p className="text-sm text-muted-foreground">  
64 274    {filteredBoard.columns.length}  
65 275    {filteredBoard.columns.reduce((total, column) => total + co  
66 276    件のタスクが見つかりました  
67 277    </p>  
68 278  })  
69 279  
70 280  <DndContext  
71 281    sensors={sensors}  
72 282    onDragStart={handleDragStart}  
73 283    onDragOver={handleDragOver}  
74 284    onDragEnd={handleDragEnd}  
75 285  >  
76 286    <div className="flex gap-6 overflow-x-auto pb-4">  
77 287    {filteredBoard.columns.map((column) => (  
78 288      <div key={column.id} className="flex-shrink-0 w-80">  
79 289        <DroppableColumn column={column} />  
80 290      </div>  
81 291    ))}  
82 292  </div>
```

Your Task: /Users [REDACTED] andbox/kanban-app/components/drag-drop-board.tsx の フィルター入力フィールドの下に、フィルタ結果の表示情報を追加してください。TODO(human) コメントの場所に実装してください。

Guidance: フィルタされたタスクの総数を表示し、フィルタがアクティブ (filterTextが空でない) な場合は「〇件のタスクが見つかりました」のようなメッセージを表示してください。filteredBoardから各columnのtasks.lengthの合計を計算することができます。フィルタが無効な場合は何も表示しないか、全タスク数を表示するかを検討してください。

- > 実装が完了しました
  - ↳ Selected 6 lines from components/drag-drop-board.tsx in Visual Studio Code
- ありがとうございます！フィルタリング結果の表示を実装していただきました。ただし、少し修正が必要ですね。現在のコードはカラム数を表示していますが、タスクの総数を表示する方が適切です。
- Update(components/drag-drop-board.tsx)

Opened changes in Visual Studio Code ⓘ  
Save file to continue...

Do you want to make this edit to drag-drop-board.tsx?  
↳ 1. Yes  
2. Yes, allow all edits during this session (shift+tab)  
3. No, and tell Claude what to do differently (esc)

## 本日の問い合わせ（再掲）

「AIに聞けば3秒で答えが出る時代に、なぜ私たちは学び続ける必要があるのか？」

## 回答

AIが生成したコードの品質を評価し、責任を持ち、応用できる「自ら考える力」が、これまで以上に重要になるから。

# まとめ

- AIの登場により知識獲得が受動的になったが、本質的理解がなければコードレビューや技術責任を負えない
- 「劇場のイドラ」を避け批判的思考力を身につけることで、AIを「答えを出す道具」から「思考を深めるパートナー」へ転換
- 学習モード活用により、ソクラテス式問答でメタ認知を養い、自分で考える力を維持できる
- 手を動かした経験は長期記憶に残るため、短期的効率性より将来への投資として学習継続が重要
- Claude Codeの学習モードを活用し、実際にコードを書くデモを行った