HANDS-ON ・ VIBE CODING

アプリ2本と、社内に広げる道筋

GenAI活用スキル向上支援 第3回｜アーサー・ディ・リトル・ジャパン株式会社 御中｜2026年8月21日（金）

正味 **[360min]**　持ち帰り **4つ**　講師 **安田 光喜**

S1 今日の問い

第1回（7月31日）で Claude Code と保管庫を、第2回（8月3日）で業務を1つ動くものにする流れを扱いました。今日は、そこで得た手触りを会社としての判断に変換します。

技術を覚えていただく日ではありません。手を動かすのは、次の4つを自分の言葉で語れるようになるためです。

今日決めに来ていること

- これを会社にどう普及させるか
- 社内で誰に、何を、どこまでやらせるか
- どうやって効果を測るか
- どうやって成果物を評価し、安全性を担保するか

持ち帰るもの4つ

終わったときに手元に残ります

- **タスク管理アプリ**　HTMLファイル1本。ダブルクリックで動きます
- **ご自身の業務アプリ**　保管庫の `50_業務改善/` の中に残ります
- **安全点検レポート**　情報システム部門に提出できる形の `点検レポート.md`
- **自部門の効果測定ボード**　施策ごとの削減時間と金額が積み上がります

今日の流れ

午前 [150min]

- [15min] 今日の問い
- [10min] 環境の確認。ファイルをダブルクリックするだけです
- [35min] 経営から見たバイブコーディング
- [10min] 休憩
- [25min] 何が作れるのか。見本3本を触ります
- [25min] AIへの指示の仕方
- [30min] 作らせてみる。投げたら昼休憩です

午後 [210min]

- [15min] 午前の回収
- [25min] 自分の業務を1つ選ぶ
- [45min] 自分の業務アプリ。待ち時間は座学に切り替えます
- [10min] 休憩
- [25min] 直す・仕上げる
- [30min] 安全に配れる状態にする
- [20min] 配る・公開する
- [20min] 効果をどう測るか
- [20min] 社内に広げる道筋

進め方の約束

3つだけ

- **待ち時間は必ず出ます。**その間は座学に切り替えます。AIが動いている画面は放っておいて構いません
- **詰まったら手を挙げてください。**同じところで複数名が詰まったら、全体を止めます
- **実データは入れません。**この1点だけは例外なしです

入れないもの

クライアント名、案件名、未公開の財務数値、個人情報、社内の未公開資料の本文、認証情報。演習で使う題材は `02_タスク管理アプリ演習/data/` にダミーを用意してあります。A社からE社はすべて架空の呼称です。

深い質問は最後に回します

座学とハンズオンを交互に置いています。座学が伸びると手を動かす時間が削られるので、踏み込んだ質問は最終セッションの質疑に送らせてください。時間は残り全部を充てます。

貴社のレポートから引きます

2026年6月17日の「AI-First or Disrupted」で、パートナーの Petter Kilefors 氏が「本当の分断は導入企業と非導入企業の間ではなく、仕事を最適化する側と仕事を消す側の間にある」と述べています。今日の6時間は、この線のどちら側に自部門を置くかを決める時間です。

S2 環境の確認

ダブルクリック1回で終わります。インストールするものはありません。

場所 **配布フォルダの一番上** 使うもの **SETUP-Mac.command ／ SETUP-Windows.bat**

成果物 **この資料がブラウザで開いた状態** 目安 **[10min]**

考える **何を確認しに行くのか**

セットアップファイルが見るのは4項目です。配布フォルダの中身がそろっているか、Claude Code が起動できるか、追加の道具（`/fix` と `/secure`）を置けたか、この資料を開けるか。欠けているものがあれば画面に「要対応」と件数が出ます。

> **今日はここで詰まりません**
> 成果物はHTMLファイル1本です。開発環境も、追加のインストールも、当日ダウンロードするものもありません。会場のネットワークが細くても止まりません。

書く **配布フォルダを、書類フォルダに置く**

受け取ったZIPを展開し、フォルダごと書類フォルダに置いてください。Mac は `~/Documents`、Windows は `ドキュメント` です。デスクトップ直下や、他の案件フォルダの中には置かないでください。

> **1案件1フォルダで起動します**
> Claude Code は起動した場所から下を読み取ります。デスクトップ直下で起動すると、隣のクライアントの資料まで読み取りの射程に入ります。運用ルールとしてはここが一番効きます。

実行 **セットアップファイルをダブルクリックする**

OS	ダブルクリックするファイル	止められたときの操作
Mac	`SETUP-Mac.command`	「開発元を検証できません」と出たら、`Control` を押しながらクリックして「開く」を選びます

Windows	`SETUP-Windows.bat`	SmartScreen の青い画面が出たら「詳細情報」から「実行」を選びます

黒い画面が開いて、1分ほどで4項目の確認が終わります。最後にこの資料がブラウザで開きます。

確かめる **「準備が終わりました」と出ているか**

画面の最後に「準備が終わりました」と出ていれば完了です。黒い画面は閉じて構いません。

「要対応が N 件あります」と出た場合は、その画面のまま手を挙げてください。何が欠けているかが画面に出ているので、その場で直します。Claude Code さえ動けば午前は最後まで通ります。

> Claude Code まわりで詰まったときは `02_タスク管理アプリ演習/hints/step5_詰まったとき.md` を開いてください。

達成チェック

- [ ] 配布フォルダを書類フォルダに置いた
- [ ] セットアップファイルをダブルクリックした
- [ ] 「準備が終わりました」を確認した

早く終わった方へ

`02_タスク管理アプリ演習/` の中を見てください。`CLAUDE.md` に「指示を出す人はプログラミングの経験がない」と書いてあります。この1行がAIの返答の文体を変えています。`.claude/settings.json` には実際の禁止設定が入っていて、`99_参考解/` への書き込みと `rm` の実行が止めてあります。

S3 経営から見たバイブコーディング

聞いていただくだけです。あとで引けるよう、数字と日付を置いておきます。

やること **聞く**　目安 **[35min]**

言葉の1年

原典は2025年2月2日（UTC。日本時間では2月3日）、Andrej Karpathy の X 投稿です。本人はこう書いていました。

```
There's a new kind of coding I call "vibe coding" ... I "Accept All" always,
I don't read the diffs anymore. ... It's not too bad for throwaway weekend
projects.
```

最初から使い捨て限定でした。Simon Willison が2025年3月19日に狭義を確定させています。「LLMが書いたコードをレビューせずにソフトウェアを作ること」。同じ記事で「LLMが書き、あなたがレビューし、十分にテストし、動作を説明できるなら、それはバイブコーディングではなくソフトウェア開発だ」とも書いています。

> **今日やるのは後者です**
> レビューして、確かめて、説明できる状態まで持っていきます。狭義のバイブコーディングは今日の対象外です。

Collins English Dictionary が2025年11月6日に vibe coding を Word of the Year 2025 に選びました。その3か月後、2026年2月4日に Karpathy 本人が言葉を降ろしています。いまの本人の呼び方は agentic engineering です。用語の賞味期限が1年、という実例です。

数字で見る現在地

出どころ	数字	読み方
Y Combinator（2025年3月、W25バッチ）	25%のスタートアップで、コードの95%がAI生成	Jared Friedman が同じ場で付けた但し書きがほぼ引用されません。採択されたのは全員が高度に技術的で、ゼロから自分で書ける人間だった。「素人でも作れる」の証拠ではなく「書ける人が書かなくなった」の証拠です

Stack Overflow Developer Survey 2025（約49,000名・177か国）	AIツール利用84%。精度を「非常に信頼する」は3.1%	信頼するという回答は前年40%から29%へ11ポイント下がりました。しかも経験の長い開発者ほど不信が強い。慣れるほど信頼が上がる普通の採用曲線と逆行しています
METR（熟練OSS開発者16名・246課題のRCT）	AI許可時の所要時間は19%長かった	事前予測は「24%速くなる」、実験後の自己評価でも「20%速くなった」。体感と実測が逆方向を向いています
IPA「DX動向2026」	個人で業務利用63.3%、部署の業務プロセスに組み込み11.9%	ツールは入ったが、業務の作り替えには入っていません。今日の主題はこの11.9%側に1件入れることです
令和8年版情報通信白書（2026年7月24日）	コード生成AIの個人利用経験率　日本7.0%、米国35.4%、中国33.5%	生成AI全体の利用経験率は日本58.8%まで上がりました。作らせる用途だけが5倍差で置いていかれています

コンサルティング業務での価値

1. **検証のための複製。**FT報道によれば、Bain は買収候補の SaaS 製品を数百本、バイブコーディングで作り直してデューデリに使っています。グローバル PE プラクティス責任者 Rebecca Burack の言葉が「2Dで見るのと3Dで見るくらいの違い」。ある PE 投資家は複製版を見てアナリティクス基盤の入札から降りました。2023年は専任エンジニアチームの仕事だったものを、いまは一般のコンサルタントが回しています
2. **提案に添える動くもの。**紙で説明する代わりに触らせる
3. **自分の反復作業を消す。**LINEヤフーは人事総務領域の10件の生成AIツールで、月間約1,600時間以上の削減を見込んでいます

うまくいかなかった例

案件	金額と分量	何が起きたか
Deloitte 豪 DEWR 向け	A$439,000・237ページ	存在しない論文、実在しない研究者への帰属、連邦裁判所判事の架空引用。報告書の一部を返金

Deloitte カナダ ニューファンドランド・ラブラドール州向け	C$1,598,485・526ページ	存在しない引用が少なくとも4件

どちらも本文の分析自体は維持されています。壊れたのは検証プロセスの側です。しかも豪の件は、AI利用そのものではなく開示がなかったことで信用が折れました。訂正版では開示を追記しています。

動くことと、正しいこと

Veracode が150超のLLMに同一課題を解かせたところ、コードの正しさは95%超なのに、セキュアなコードが出る割合は約55%でした。2年間のモデル進化で、この数字は動いていません。

```
Models have become excellent at writing code that compiles.
They've failed at writing code that's safe.
```

デモで動くことは受け入れ基準になりません。ここが今日いちばん重い一文です。

社内ツールだから安全、は成り立ちません

セキュリティベンダーのブログ（Treblle、2026年3月18日）によれば、CodeWall の自律型攻撃AIエージェントが、McKinsey の社内ツール Lilli に対して2時間で22の未認証APIエンドポイントを発見し、4,650万件のチャットメッセージへ到達できる状態だったとされています。McKinsey・CodeWall いずれの一次発表も確認できていないため、この件は「ベンダーの検証報告」として扱ってください。外に出さないから安全、という前提が置けないことは、それでも変わりません。

機密情報の扱い

パートナーが知っておくべき事実3つ

- **契約プランで行き先が変わります。**Free/Pro/Max は学習許可をオンにすると非識別化して最大5年保持、オフなら30日。商用契約（Team/Enterprise/API）は既定で学習不使用・30日です
- `.gitignore` と `.claudeignore` **は読み取り制限になりません。**The Register が2026年1月28日に実証しています。`.env` を普通に読んでコンソールに出しました。効くのは `settings.json` の `permissions.deny` だけです
- **会話は手元に残ります。**セッションの全会話が `~/.claude/projects/` に平文で30日ローカル保存されます。私物PCで使う場合、この残り先を会社が消去できません

私物PCの正しい言い方

「私物PCだから社内システムに届かず安全」ではありません。正しくは「私物PCだから統制が効かない。対象者・期間・扱ってよいデータ区分・終了後の消去手順を決めて、期限付きで受容する」です。シャドーITとの差は技術ではなく、承認記録と期限の有無だけです。

S4 何が作れるのか

言葉で説明するより、触っていただくほうが速く済みます。動く見本を3本入れてあります。

場所 **06_見本アプリ**　使うもの **ブラウザ**

成果物 **3本とも触って、作りたいものの当たりを付けた状態**　目安 **[25min]**

考える　**この形で足りる範囲**

今日作るものは、すべてHTMLファイル1本です。ダブルクリックするとブラウザが開きます。インストールも起動コマンドも要りません。外部と通信せず、入力したデータはそのブラウザの中だけに残ります。

この形が向くもの

- 自分か、数人が使う道具
- 入力して、集計して、並べ替えて、見る
- Excel でやっているが、毎回同じ手順を踏んでいる作業

この形では足りないもの

- 複数人が同じデータを同時に見る、書き換える
- 社内システムのデータを直接読む
- 権限で見える範囲を分ける

足りなくなった時点で、初めて別の作り方を検討します。最初からその心配をすると、1本も作らないまま終わります。

書く　**見本3本を開く**

`06_見本アプリ/` を開いて、3本ともダブルクリックしてください。ブラウザのタブが3つ開きます。

ファイル	何をするものか	今日どこで使うか
`01_効果測定ボード.html`	部署ごとの施策を1行ずつ足すと、月あたりの削減時間と金額が積み上がる	効果をどう測るかのセッションで、自分の施策を入れます

`02_重み付け比較表.html`	選択肢を評価軸ごとに5段階で採点し、軸の重みを掛けて総合点を出す。重みを動かすと順位が変わる	持ち帰り。ベンダー選定や投資判断の議論に使えます
`03_貼るだけ集計ボード.html`	Excel の範囲をコピーして貼ると、列を選ぶだけで集計とグラフが出る	持ち帰り。貼ったデータは外に出ません

実行 **1本は、実際にデータを入れて動かす**

眺めるだけでは判断材料になりません。`02_重み付け比較表.html` か `03_貼るだけ集計ボード.html` のどちらかで、実際に手元の数字を入れてみてください。ダミーで構いません。

入れたあとブラウザを閉じて、もう一度開いてください。入力が残っています。これが localStorage です。サーバーもデータベースも使わずにデータが残る仕組みは、これだけです。

> **残る場所は1か所だけ**
> データは、いま使っているPCの、いま使っているブラウザの中にだけ残ります。別のPCで同じファイルを開いても、入れたデータは出てきません。この性質が、情報システム部門への説明で最も強い材料になります。

確かめる **カタログから、自分の候補を2つ選ぶ**

`06_見本アプリ/カタログ.md` に、この形で作れるものが70件並んでいます。実際に作られて使われている社内ツールから引いたものです。

上から順に読んで、自部門で使えそうなものに2つだけ印を付けてください。1つに絞らなくて構いません。午後のワークシートで使います。

> **選ぶときの基準**
> 「効果が大きそうなもの」ではなく「自分が今週やる作業」を選んでください。効果の大きさは、使われ続けたかどうかでしか決まりません。使われないものの効果はゼロです。

自分で考えるポイント

- 3本の見本のうち、自部門にそのまま置けるものはどれか。置けない理由があるとしたら、それは機能の不足か、運用の問題か
- カタログで選んだ2つは、誰が使うものか。自分だけか、部下も使うか

達成チェック

- [ ] 見本3本をブラウザで開いた
- [ ] 1本にデータを入れ、閉じて開き直して残ることを確認した
- [ ] カタログから候補を2つ選んだ

詰まったときの逃げ道

ファイルをダブルクリックしても開かない場合は、ブラウザを先に起動してから、ファイルをブラウザの窓にドラッグしてください。それでも開かなければ手を挙げてください。

S5 AIへの指示の仕方

実演です。同じ依頼を4段階で投げて、出てくるものの差を見ていただきます。

やること **見る**　目安 **[25min]**

4段階

段階	指示のしかた	何が変わるか	どこに置くか
1	思いついたまま話す	動くものは出る。毎回違うものが出る	どこにも残らない
2	型に沿って書く	狙ったものが出る確率が上がる	手元のメモ
3	型そのものを道具にする	毎回同じ型で出る。人に渡せる	`.claude/skills/`
4	道具を並べて流す	手順ごと再現できる。分担できる	Skill ＋ サブエージェント ＋ フック

経営として見るべきは段階3です。ここで初めて、個人の技が組織の資産になります。段階2で止まっている限り、うまい人の手元にしか残りません。IPA が示す63.3%と11.9%の差は、この段の差です。

段階2の型は4つ。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。`02_タスク管理アプリ演習/要件メモ.md` がこの型そのものになっています。

2026年に変わったこと

カスタムコマンドが Skill に統合されました。公式ドキュメントの表現は「Custom commands have been merged into skills」。`.claude/commands/deploy.md` と `.claude/skills/deploy/SKILL.md` はどちらも `/deploy` を作ります。同名なら Skill が勝ちます。

もう1つ。Agent Skills 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。

ロックインが怖いという質問への答え

モデルとエディタには賭けず、資産の形式に賭けます。Amazon Q Developer は2027年4月30日で終わりますし、Cursor は SpaceX による買収が2026年6月16日に合意されました。プロダクトの寿命は読めません。書いた指示が持ち運べるかどうかで選びます。

プランモードと仕様駆動

`Shift` + `Tab` でプランモードに切り替えます。読むだけで編集させず、計画を出させて、合意してから実装に入ります。

ただし Anthropic 公式が同時にこう書いています。「If you could describe the diff in one sentence, skip the plan.」判断軸は案件の重要度ではなく、不確実性の量です。

反対側も出します。Thoughtworks の Birgitta Böckeler は仕様駆動の主要ツールを実際に検証したうえで推奨していません。理由は、問題の大小に関わらず同じ手順を強いる硬直性、コードレビューより重い文書レビューの負荷、精緻な仕様を書いても指示が無視される「コントロールの錯覚」の3点です。記事はドイツ語の Verschlimmbesserung（改善のつもりの改悪）で締めています。

今日の立場

不確実性が高いものだけ計画します。全部に仕様を書くと、書く手間が回収できません。

禁止と統制は別物

ここは強く言います

`CLAUDE.md` に「.env を編集するな」と書いても、それは要望にすぎません。`PreToolUse` フックと `permissions.deny` だけが強制になります。社内規程を配っただけで統制した気になる構造と、まったく同じ話です。

S6 作らせてみる

手を動かします。要件を自分の手で書き、道具を組ませ、実装を投げてから昼休憩に入ります。

場所 **02_タスク管理アプリ演習**　使うもの **要件メモ.md ／ /pipeline ／ /build**

成果物 **仕様書.md と、動きはじめた app.html**　目安 **[30min]**

考える **誰が、いつ、何を見たいのか**

大きく考えると手が止まります。「案件管理システム」ではなく、それを使う瞬間まで下ろしてください。

1. それを使う瞬間はいつですか　→「月曜の朝、今週やることを確認するとき」
2. そのとき何を見たいですか　→「今週締切のものと、誰が持っているか」
3. いまそれをどこで見ていますか　→「Excelとメールとチャットに散らばっている」

3まで下りると「今週締切のタスクを、担当ごとに1画面で見る」になります。これなら今日のうちに動きます。

書く **要件メモ.md に、自分の手で書く　[10min]**

`02_タスク管理アプリ演習/要件メモ.md` を開いて、欲しい機能を箇条書きで10行書きます。**ここだけはAIに書かせません。**きれいに書く必要はありません。「締切が近い順に並んでほしい」「終わったものは薄く表示したい」で十分です。

埋める欄

- 誰が使うか
- 何を管理したいか
- 欲しい機能（10行）
- できたと言える条件（3つまで）
- やらないこと（2つまで）

完成の条件だけは丁寧に

「見やすくする」は完成しません。判定できません。「タスクを10件入れても1画面に収まり、横スクロールが出ない」なら、できたかどうかが目で分かります。数字か◯×が付く形にしてください。部下に仕事を渡すときと同じです。

何を書けばいいか分からないときは `hints/step1_要件メモ.md` を開いてください。埋めた例が丸ごと1本入っています。

実行 **道具を組み立てさせて、実装を投げる　[15min]**

演習フォルダで Claude Code を起動し、次を送ります。

```
/pipeline
```

要件メモを読んで、このアプリ専用の道具を `.claude/` の中に作ります。仕様をまとめる係、実装する係、点検する係。画面上でAIが自分の道具を書いていくところが見えます。

> **ここが今日いちばんの見どころです**
> 指示の型の段階3と段階4が、その場で目の前で組み上がります。ファイルを作り、フォルダ構造を持ち、次のセッションでその道具を呼び出せる。ブラウザのチャットでは、ここに届きません。

途中で最大3つまで質問されます。答え終わると `仕様書.md` ができて、`/build` と `/check` が使えるようになります。そこまで来たら実装を投げます。

```
/build
```

5分から10分かかります。**投げたら昼休憩です。**画面は放っておいて構いません。

> `/pipeline` が候補に出ない、質問が多すぎて進まないときは `hints/step2_pipeline.md` を開いてください。

確かめる **投げる前に、仕様書の2か所だけ見る**

`仕様書.md` の全文は読まなくて構いません。`## 作るもの` と `## 完成の条件` の2か所だけ見てください。

- 作るものが1文で書かれていて、自分の意図と合っているか
- 完成の条件が、判定できる形になっているか

違っていたら `仕様書.md` を直接書き換えて、「仕様書.md を書き換えました。読み直して、道具を作り直してください」と伝えます。

自分で考えるポイント

- 自分が書いた10行のうち、AIが仕様書に落とせなかったものはどれか。落とせなかった理由は何か
- 「完成の条件」を書き換えられたなら、元の書き方の何が判定できなかったのか

達成チェック

- [ ] 要件メモ.md に、自分の手で機能を書いた
- [ ] 完成の条件を、数字か◯×が付く形で書いた
- [ ] /pipeline を実行し、仕様書.md ができた
- [ ] /build を投げてから昼休憩に入った

詰まったときの逃げ道

時間内に `/build` まで届かなくても構いません。午後の最初で回収します。要件メモが書けた時点で、今日の主目的の半分は済んでいます。

S7 午前の回収

動かします。「動いた」と「条件を満たした」が別物であることを、その場で見ていただきます。

場所 **02_タスク管理アプリ演習**　使うもの **app.html ／ /check ／ /fix**

成果物 **ブラウザで動く app.html**　目安 **[15min]**

考える　**何をもって「できた」と言うつもりだったか**

開く前に、午前に書いた `要件メモ.md` の「できたと言える条件」を読み返してください。3つあるはずです。この3つが、これから判定の物差しになります。

書く　**app.html をダブルクリックする**

演習フォルダに `app.html` ができています。ダブルクリックすると、いつも使っているブラウザで開きます。起動コマンドはありません。

開いたらタスクを3件ほど入れてみてください。入れたあと、ブラウザを閉じてもう一度開いて、残っていることを確かめます。

実行　**点検させて、直す**

```
/check
```

完成の条件を1つずつ実際に確かめて、表で報告が出ます。満たしていない条件が1つは出ます。出たら直します。

```
/fix
```

直したいところを伝えると、現状・達したい状態・制約の3点に整理してから直します。整理された3点が画面に出るので、意図と合っているかをそこで確認してください。

確かめる　**作り変えの指示を1回入れる**

機能を足すのではなく、並べ方を変えます。作り直しではなく作り変えができることを、ここで見ます。

Claudeへの指示（コピーして送る）

締切が近い順ではなく、案件ごとにまとめて表示してください。
案件の中では締切順のままにしてください。

直したあと、ブラウザで `Command` + `R`（Windows は `Ctrl` + `R`）を押して読み込み直します。一から作り直されていないことを確かめてください。

エラーが出た、画面が真っ白になったときは `hints/step3_build.md` を開いてください。

自分で考えるポイント

- `/check` が「足りない」と言った条件は、なぜ実装から漏れたのか。仕様書の書き方の問題か、条件の書き方の問題か
- 作り変えの指示で、頼んでいない場所まで変わっていないか

達成チェック

- [ ] app.html をダブルクリックして開いた
- [ ] /check を実行し、満たしていない条件を見つけた
- [ ] /fix で直した
- [ ] 並べ方を変える指示を1回入れた

詰まったときの逃げ道

`/build` が完走していない方は、まず「いまどこまで進んでいますか。残りを続けてください」と送ってください。それでも動かない場合は `99_参考解/app_完成版.html` に完成形が入っています。研修中は開かない前提ですが、追いつけないときは講師の合図で開きます。

S8 自分の業務を1つ選ぶ

午後に作るご自身の業務アプリの下書きです。入り口は2つあります。どちらを選んでも構いません。

場所 **03_ワークシート**　使うもの **AI活用ワークシート_バイブコーディング版.xlsx**

成果物 **次のセッションに貼り付ける下書き**　目安 **[25min]**

考える　**どちらの入り口から入るか**

入り口A　新しく書く

- シート「① 候補を出す」で候補を5つ書き、1つ選ぶ
- シート「② 何を作るか」を埋める

入り口B　いまのワークシートを作り替える

- 第2回（8月3日）に書いたシートを Claude Code に渡す
- 「AIに何を聞くか」を書く形から、「AIに何を作らせるか」を書く形へ作り替えさせる

入り口Bは、道具そのものをAIに直させる体験になります。配られたものを埋めるだけの紙ではなく、自分の仕事に合わせて作り替えられる。この感覚がここで手に入ります。

書く　**直近1か月で、面倒だと思った作業を1つ**

小さなことで構いません。むしろ小さいほうが今日のうちに動きます。`06_見本アプリ/カタログ.md` で印を付けた2つがあれば、そこから選んでも構いません。

- 複数のファイルから数字を集めて1つの表にまとめる
- 決まった形式の資料を、毎回同じ手順で作る
- 受け取った文書を読んで、必要な部分を別の場所に転記する
- 前年や前月の資料を探して、数字だけ差し替える

完成の条件の書き方

シート「③ 完成の条件」に、終わらない言い方と判定できる言い方の対応表が入っています。「見やすくする」は完成しません。「1画面に収まる」なら完成したか判定できます。

実行

入り口Bを選んだ方は、これを送る

Claudeへの指示（コピーして送る）

```
このワークシートは「AIに何を聞くか」を書く形になっています。
「AIに何を作らせるか」を書く形に作り替えてください。
いま書いてある内容は捨てずに、新しい形に移してください。
足りない項目があれば、私に質問してください。
```

確かめる

誰が使うかを1行で決める

埋まっていない欄があっても進んで構いません。ただし1つだけ決めてください。

これは自分だけが使うものか、部下も使うものか。

作った本人しか使っていないツールの比率は、効果測定で最初に見る数字です。自分だけ、と決めたなら、それはそれで正しい判断です。決めていないまま作ると、あとで誰も測れません。

自分で考えるポイント

- 選んだ作業のうち、判断が入る部分はどこか。その判断は条件として書き出せるものか
- 入り口Bを選んだ方は、AIが「足りない」と指摘してきた項目は何か。それは元のワークシートの欠落か、ご自身の業務の特殊事情か

達成チェック

- [ ] 対象の作業を1つに絞った
- [ ] 完成の条件を、判定できる形で書いた
- [ ] 自分だけが使うか、他の人も使うかを決めた

詰まったときの逃げ道

題材が決まらない方は、午前に作ったタスク管理アプリをご自身の部門向けに作り替える、でも構いません。ゼロから考えるより速く動きます。

S9 自分の業務アプリ

今日の本番です。前半は壁打ち、後半は待ち時間になります。待ち時間は座学に切り替えます。

場所 **Claude Code（PMVault で起動）** 使うもの **/gyomu**

成果物 **50_業務改善/ の中に設計書と app.html** 目安 **[45min]**

考える **どこで起動するか**

PMVault で Claude Code を起動します。Skill はユーザー設定に置いてあるのでどのフォルダでも呼べますが、保管庫の中で起動すると成果物の置き場所を聞かれずに進みます。

> **実データは入れません**
> ご自身の業務が題材ですが、クライアント名と未公開の数値は書かないでください。「A社」「約N億円」に置き換えても、要件の形は変わりません。

書く **ワークシートの内容を、そのまま貼る**

Claudeへの指示（自分の内容に書き換えて送る）

```
/gyomu

（ワークシートに書いた内容をここに貼る）

成果物はHTMLファイル1本にしてください。
外部のライブラリを読み込まず、データはブラウザの中に保存してください。
```

最後の2行が効きます。これを書かないと、環境の用意が要るものを作られることがあります。

実行 **壁打ちに答える　[15min]**

1回に1問か2問、質問が返ってきます。答えていくと、数往復で要件が固まります。要件が固まると設計書が書かれ、承認を聞かれます。

> **ここが山場です**
> 「経験です」「勘です」と答えると、そこで終わりません。「先月やったときはどうしましたか」「その前の月は」と、過去の実物を1件ずつ聞いてきます。面倒に感じるところですが、3件

並べるとだいたいルールの形になります。判断を言語化する作業そのものが、今日の持ち帰りです。

承認したら実装に入ります。ここから待ち時間です。画面は放っておいて構いません。

確かめる **待ち時間に、普及の型と失敗の型を聞く　[30min]**

座学に切り替えます。いまご自身のPCで起きていることの説明として話します。

普及の型は3つしかありません

- **CoE方式**　中央に推進組織を置き、基準と教材を配る。速いが、現場のニーズから離れると死蔵される
- **段階展開方式**　1部門で業務プロセスに埋めてから横に広げる。IPA が言う11.9%側に入れる型はこれ
- **チャンピオン方式**　各部門に旗振り役を1人置く。人に依存するので、異動で消える

IPA「DX動向2026」では、経営者・IT部門・業務部門の協調が「十分＋まあまあできている」と答えた比率は、DX成果が出ている企業で72.9%、出ていない企業で37.1%。倍近い開きがあります。号令だけでも、現場任せでも動きません。

パイロットから本番へ上げる6条件

- 作った本人以外が、説明なしで1回使えた
- 停止しても業務が止まらない、または代替手順が文書化されている
- 入力データの機微区分が判定済みで、外部へ出る範囲が特定されている
- 誤りが出たときに責任を持つ人が、1人の名前で決まっている
- 保守する人が決まっていて、その工数が別の仕事から引かれている
- 削減時間または品質の変化が、導入前と比較できる形で1回測られている

4番目と5番目が抜けたまま横展開した結果が、Bain の19,000本のカスタムGPTのロングテールであり、日本企業の11.9%です。

自分で考えるポイント

- 質問に答えていくうちに、自分が何を判断していたのかが言葉になった瞬間はどこか
- いま作っているものを本番に上げるとして、6条件のうち何番が埋まっていないか

達成チェック

- [ ] /gyomu にワークシートの内容を貼って送った
- [ ] 壁打ちに答え、設計書の承認まで進んだ
- [ ] 50_業務改善/ の中に成果物ができた

詰まったときの逃げ道

途中で止まったように見えたら、まず5分待ってください。複数の係が同時に動いているため、画面に何も出ない時間があります。それでも動かない場合は「いまどこまで進んでいますか」と聞けば状況が返ります。時間内に実装まで届かなくても、要件が言葉になった時点で持ち帰る価値があります。

S10 直す・仕上げる

「ここが変」だけでは直りません。3点で書くと1回で直ります。見た目の決めごとは、言い直す代わりにファイルに置きます。

場所 **作業中のアプリのフォルダ**　使うもの **スクリーンショット ／ /fix ／ DESIGN.md**

成果物 **直って、整った app.html**　目安 **[25min]**

考える　3点に割り振る

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」と出したい。過ぎているものは「2日超過」
制約	触ってほしくない場所、変えてほしくない部分	列の並びは変えない。色は増やさない

制約が一番効きます

制約を書かないと、頼んでいない場所まで直されます。直ったが別のところが壊れた、が一番時間を溶かします。

書く　画面を撮って、貼る

撮り方

- Mac　`Shift` + `Command` + `4` で範囲を選ぶ
- Windows　`Windows` + `Shift` + `S` で範囲を選ぶ

貼り方でつまずきます

Claude Code の入力欄では、Mac も Windows も `Ctrl` + `V` を使います。Mac で `Cmd` + `V` が効くのは iTerm2 だけです。うまく貼れないときは、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 のドキュメントに「Claude works best when images come before text」とあります。画像を先に貼って、そのあとに言葉を足してください。

実行 **同じ不具合を2通りで伝えて、差を見る**

まず、雑に伝えます。

1回目（わざと雑に）

ここが変です。直してください。

次に、3点で伝えます。

2回目（3点セット。自分の状況に書き換える）

現状	締切の列が日付だけで、あと何日あるか分からない
達したい	締切の右に「あと3日」と出したい。過ぎているものは「2日超過」
制約	列の並びは変えない。色は増やさない

通じない言い方と通じる言い方の対応表が5組、`hints/step4_直す指示.md` に入っています。

確かめる **見た目の決めごとを、ファイルに置いて一度だけ言う**

AIに作らせると全部同じ見た目になります。紫のグラデーションが出る理由は美意識ではなく統計です。Web の中央値が訓練データを飽和させた結果、そのまま返ってきます。避けた先も、次の中央値になります。逃げ切りは構造的にありません。

だから自分の側から決めます。`02_タスク管理アプリ演習/DESIGN.md` を開いて、書き換えてください。書く項目は6つです。

DESIGN.md に書く6項目

- 色　… 2色まで。3色以上を使うと、それだけで素人の作ったものに見えます
- 文字　… 本文のサイズと行間。見出しは本文の1.2倍まで
- 余白　… カードの内側、要素の間、ページの左右
- 角と影　… 角丸は何pxまでか。影を使うか
- 動き　… どこで何秒動かすか
- やらないこと　… ここが一番効きます

Claudeへの指示（コピーして送る）

```
DESIGN.md に従って、画面全体の見た目を整えてください。
機能は変えないでください。
画面幅390pxで横スクロールが出ないようにしてください。
```

整ったら、ブラウザの窓をスマートフォンくらいの幅まで細くしてください。横スクロールバーが出なければ通っています。

言葉を1つ知っているだけで結果が変わるもの

頼み方の語彙は、ワークシートのシート「④ 頼み方の語彙集」と `04_表現テンプレ集/` に入れてあります。読み上げません。手元で引けることだけ確認してください。

やりたいこと	開くファイル
処理の流れを図にしてほしい	`04_表現テンプレ集/01_図解の頼み方.md`
画面の部品を名前で指定したい	`04_表現テンプレ集/02_画面の部品の呼び方.md`
動きと見た目を数字で指定したい	`04_表現テンプレ集/03_動きと見た目の指定.md`
人に渡す形を決めたい	`04_表現テンプレ集/04_人に渡す形.md`

実務で一番効く1語

エンプティステート。データが0件のときに何を出すかを指定しないと、AIは空白の画面を作ります。「まだ登録がありません。右上の追加から始めてください」まで書かせてください。渡した相手が最初に見る画面が、これです。

自分で考えるポイント

- 1回目と2回目で、返ってきたものはどう違ったか。1回目は何を勝手に決めたか
- `DESIGN.md` の「やらないこと」に何を書いたか。書かなかったら、AIは何をしていたか

達成チェック

- [ ] スクリーンショットを撮って貼り付けた
- [ ] 同じ不具合を、雑な伝え方と3点セットの両方で試した
- [ ] DESIGN.md を自分の決めごとに書き換えて整えさせた
- [ ] 画面幅390pxで横スクロールが出ないことを確認した

詰まったときの逃げ道

整えさせたら機能が壊れた場合は、`Esc` を2回押して巻き戻してください。直近100個、30日ぶんが残っています。巻き戻したあと、「見た目だけを変えてください。動きのコードには触らないでください」と制約を足して投げ直します。

S11 安全に配れる状態にする

作ったものを社内で使う前に通します。出てくるのは、情報システム部門にそのまま出せる報告書です。

場所 **作業中のアプリのフォルダ** 使うもの **/secure** 成果物 **点検レポート.md** 目安 **[30min]**

考える **審査は4点しか聞かれません**

情報システム部門が見ているのは「ツールが安全か」ではありません。誰の権限で、何に到達し、どこに残り、後から追えるか。この構造に落とすと、質問は4つになります。

この4つに答えられれば通ります

- どこに何を送るか
- どこに何を保存するか
- 誰が見られるか
- やめたいときにどう消すか

単一のHTMLファイルで、外部と通信せず、データがブラウザの中だけに残る形であれば、この4つはすべて「外に出ない」で埋まります。今日この形を選んだ理由の3分の1が、これです。

逆効果になる説明

「エンタープライズグレードなので安全です」「リスクはゼロにできます」は、審査側の役割を否定する言い方なので必ず止まります。Anthropic が2026年7月17日に公開した CISO 向けガイドの表現を借りると、仕事は「ゼロリスクにすること」ではなく「リスクを可視化して境界を引き、組織として意図的に受容できる状態にすること」です。

書く

対象のファイルがある場所で起動する

点検したいアプリのフォルダで Claude Code を起動してください。午前のタスク管理アプリなら `02_タスク管理アプリ演習`、午後の業務アプリなら PMVault の `50_業務改善/` の中です。

1つのフォルダに複数のアプリがある場合は、どれを見てほしいかを先に伝えます。

実行

点検させる

```
/secure
```

チェックリストの項目を1つずつ実際に検索して確かめ、結果を3段階に分けます。「直してから使う」「承知のうえで使う」「問題なし」。「直してから使う」が1件でもあれば、まずそれが画面に出ます。

終わると `点検レポート.md` ができます。専門用語を使わず、情報システム部門の担当者が読む前提で書かれています。

勝手に直しません

「直してから使う」の項目には、直し方が1行ずつ添えられます。実際に直すかどうかは聞かれます。判断はご自身でしてください。

確かめる

レポートの第2章の表だけ読む

全文を読む必要はありません。情報システム部門への回答をまとめた表が入っているので、そこだけ見てください。自分の口で説明できる内容になっているかを確かめます。

参考として、記入例を `02_タスク管理アプリ演習/99_参考解/点検レポート_見本.md` に入れてあります。どこまで具体的に書かれていれば通るのか、粒度の目安になります。

この報告書の限界を先に言います

作ったAIが、自分の成果物を点検しています。人の目が1回も入っていません。レポートにもその旨が明記されます。社内で本格的に使う前に、情報システム部門の方に1度目を通していただく前提で扱ってください。限界を自分から先に出すと、審査側の探索コストが下がって話が速くなります。

自分で考えるポイント

- 「承知のうえで使う」に並んだ項目のうち、自部門では受け入れられないものはどれか
- この報告書を持って情報システム部門に行くとして、最初に返ってくる質問は何か

達成チェック

- [ ] /secure を実行した
- [ ] 点検レポート.md ができた
- [ ] 4点の回答を、自分の言葉で説明できることを確かめた

詰まったときの逃げ道

`/secure` が候補に出ない場合は、セットアップファイルのダブルクリックが完了していない可能性があります。もう一度実行してから、Claude Code を起動し直してください。

S12 配る・公開する

実演です。渡し方の3択を見てから、URLが出るまでを通しで見ていただきます。

やること **見る** | 使うもの **Cloudflare Workers** | 目安 **[20min]**

渡し方の3択

渡し方	誰が見られるか	向いている場面	気をつけること
ファイルを送る	送った相手だけ	最初の1人に渡すとき	相手のPCに残る。更新のたびに送り直す
共有フォルダに置く	部署の中	数人で使うとき	どれが最新版か分からなくなる。ファイル名に日付を入れる
URLで公開する	リンクを知る全員	広く見せるとき	認証が無ければ、それは公開である

最初はファイルで渡すのが安全です。使われることが分かってから、置き場所を考えます。順番を逆にすると、誰も使わないURLの管理だけが残ります。

デモの流れ

1. Cloudflare アカウントを作る（メール認証）
2. API トークンを発行する。権限は「Edit Cloudflare Workers」テンプレート。**トークンは発行時に一度しか表示されません**
3. トークンを環境変数に置く。`CLOUDFLARE_API_TOKEN` は保存済みの OAuth 認証より優先されます
4. Claude Code に「公開してください」と頼む
5. URL が出る

Cloudflare 公式が新規プロジェクトに Workers を指定しています。「If you are starting a new project, use Workers instead of Pages.」HTMLファイル1本なら、無料枠の中で収まります。

ここは見ていただくだけです

15名が同時にアカウント作成と認証を行うと、認証メールの遅延とネットワークで確実に時間を溶かします。希望される方には手順書をお渡しし、終了後に個別に対応します。

公開する前に確認すること

3つだけ

- 画面に出ている数字は、公開してよいものか
- URL を知った人は誰でも見られる。認証を付けていないなら、それは公開である
- 消したいときにどうやって消すかを、公開する前に確認したか

S13 効果をどう測るか

見本の効果測定ボードに、ご自身の施策を入れます。数字が出た瞬間に、測り方の落とし穴が見えます。

場所 **06_見本アプリ** 使うもの **01_効果測定ボード.html**

成果物 **自部門の施策が入ったボード** 目安 **[20min]**

考える **削減時間は、どこへ行くのか**

DORA が2026年3月10日に出した「Balancing AI tensions」に、そのままの一文があります。

```
the time saved during initial code or content generation is often
re-allocated to verification overhead and prompting overhead
```

浮いた時間は、検証とプロンプト調整に吸われます。同じ記事に現場の声も並んでいます。「コードを書く時間は減ったが、AIのお守りと、AIが何をやろうとしているかのレビューに時間を使っている」。

時間削減が売上に転換しない理由は3つで説明がつきます。浮いた時間が検証に吸われる。浮いた時間が別の価値ある仕事に再配置される経路を、誰も設計していない。速く作るほどレビュー待ちが伸び、律速がそちらに移る。

やってはいけない測り方

AIの利用量で社内ランキングを作る企業が実在します。DORA は2026年6月2日の記事でこれを lines-of-code と同じ虚栄指標だと切り捨てました。不要な処理を回して水増しできること、コストが10倍になってもスループットの伸びは限界的なこと、質の低い成果物を受け入れて負債が積み上がることを挙げ、Goodhart の法則を明示的に引いています。個人ランキングは、DX Core 4 も DORA も明示的に禁じています。

書く

ボードに、自分の施策を1行入れる

`06_見本アプリ/01_効果測定ボード.html` を開いて、今日作ったアプリを1行足してください。入力するのは、部署、施策名、対象人数、1回あたりの削減分、月の実施回数、時間単価です。

入れると、月あたりの削減時間と金額が積み上がります。数字が出たところで、次の確かめに進みます。

実行

その数字を、疑う

LINEヤフーは2025年7月14日に「3年で業務生産性2倍」を発表しています。ただし対象は「従業員業務の3割」と明記されています。3割の2倍なので、全体では最大でも1.18倍です。

この発表が誠実なのは、分母を書いている点です。分母を書かない「生産性2倍」は読む価値がありません。ご自身がいま入れた数字の分母は何ですか。

入れた数字を、この4点で削る

- 削減率は自己申告か、実測か。実測なら1回だけ着手から完了までを測る
- 時間単価を実勢で置くと、分子が跳ねて何でも黒字になる。低めに固定して比較可能性を優先する
- 3か月後も使われている比率（稼働継続率）を掛ける。この項がないROIは粉飾になる
- レビューと検証の工数を、独立した引き算の項として立てる。作成30分・レビュー2時間なら、それは効率化ではない

確かめる

失敗を検知する指標を、3つ決める

作った数を数える組織は必ず失敗します。見るべきは生存側です。

指標	測り方	何が分かるか
90日生存率	作成から90日後に、週1回以上使われているツールの比率	作りっぱなしになっていないか
単一利用者率	作った本人しか使っていないツールの比率	個人の効率化どまりで、組織の資産になっていない

最終更新からの経過日数	90日超で棚卸し対象、180日超で廃止候補	誰も面倒を見ていないものが何本あるか

貴社の環境ではツール側の利用ログが取れません。個人PCで、社内ネットワークにつながっていないからです。だから四半期ごとの棚卸し申告に寄せます。設計のコツが1つあって、「使っているものを申告する」ではなく「使っていないものを申告したら褒める」にすると、廃止が進みます。

先に測る対象を決める

IPA「DX動向2026」では、DXの成果指標を設定している企業は23.9%。設定しない理由の最多は「評価指標の設定方法がわからない」で36.0%、2024年度の28.7%から約7ポイント悪化しています。最初の30日は作ることではなく、どの業務のどの時間を測るかを決めることに使ってください。

自分で考えるポイント

- ボードに入れた数字のうち、実測に基づくものはいくつあるか
- 3か月後、何本が生き残っていたら成功と言うつもりか。その数字を今日決められるか

達成チェック

- [ ] 効果測定ボードに自分の施策を1行入れた
- [ ] 入れた数字の分母を確かめた
- [ ] 失敗を検知する指標を3つ決めた

詰まったときの逃げ道

入れる数字が思いつかない方は、午前に作ったタスク管理アプリを1行入れてください。対象は自分1人、削減は1回5分、実施は週5回で構いません。桁の小ささを見ることに意味があります。

S14 社内に広げる道筋

30日、90日、180日で何を置くか。誰から始めて、いつやめるか。残り時間は質疑に充てます。

場所 **07_社内で使う** やること **読む ／ 議論する** 目安 **[20min]**

最初の180日

期間	置くもの	根拠
1日目から30日	作らない。どの業務のどの時間を測るかを決める	成果指標を設定している企業は23.9%。測る対象が無いまま作ると、続けるか捨てるかを誰も判断できません
31日から60日	1部門で、業務プロセスに1件埋める	「部署の業務プロセスに組み込まれている」は11.9%。ここに1件入れることが、63.3%側との差になります
61日から90日	横展開の可否を判定する	本番へ上げる6条件で判定します。特に、責任者と保守担当が名前で決まっているか
91日から180日	棚卸しと廃止を1回回す	90日生存率と単一利用者率を出します。増やす仕組みより先に、減らす仕組みを回した組織が残ります

この順番で一番落としやすいのが最初の30日です。作りたくなるからです。ただ、測る対象を決めずに作ったものは、3か月後に「なんとなく便利だった」で終わります。

誰から始めるか

普及の型は3つあります。中央に推進組織を置く型、1部門で埋めてから広げる型、各部門に旗振り役を1人置く型。貴社の制約（個人PC、社内ネットワーク非接続、利用できるAIが限られる）では、動かせる変数がツール選定ではなく3つに絞られます。対象業務の選定、データの持ち出し可否の線引き、成果指標。この3つだけです。

だから2番目の型を推します。1部門で1件、業務プロセスに埋める。それが動いたら、同じ業務を持つ隣の部門に渡す。渡した相手が、説明なしで1回使えたかどうかで判定する。

渡すときに壊れる場所

1部署で回っていたツールを隣に渡すと、参照するデータの粒度・命名・更新頻度が違って動きません。「作ったのに広がらない」の技術的な正体はここです。渡す前に、相手の手元のデータで1回動かしてください。

やめどきを、作る時点で決める

作ったものは増えます。増えたものは、いずれ誰も面倒を見なくなります。RPAの野良ロボットで起きたことが、そのまま起きます。

対策は1行です。作った時点で「いつ見直すか」を決めておく。3か月後に、使われていなければ消す。この一言があるかないかで、1年後の棚卸しの重さが変わります。

やめる判断を、誰が持つか

- 作った本人が異動したら、そのツールはどうなるか
- 誤りが出たときに責任を持つ人は、名前で決まっているか
- 保守する工数は、誰の何の仕事から引かれているか

3つとも埋まらないなら、そのツールは個人の道具のままにしておくのが正解です。組織の資産にしようとした瞬間に、保守と責任の議論が発生します。

持ち帰る資料

`07_社内で使う/` に、そのまま社内で使える資料を4本入れてあります。研修中は開かなくて構いません。

ファイル	いつ使うか
`07_社内で使う/01_情報システム部門への説明.md`	点検レポートを持って相談に行く前に読みます
`07_社内で使う/02_効果の測り方.md`	先行指標と遅行指標を分けて、経営会議に1枚で出すとき
`07_社内で使う/03_社内に広げる道筋.md`	30日・90日・180日の設計を自部門に当てはめるとき
`07_社内で使う/04_社内ルールのひな型.md`	誰が何をどこまで作ってよいかを決めるとき

社内に持ち帰るときに考えることは `02_タスク管理アプリ演習/hints/step6_社内で使う.md` にもまとめてあります。

質疑

残り時間は全部こちらに充てます。今日踏み込めなかった論点でも、自社に当てはめたときの詰まりどころでも構いません。

CHECK 理解度チェック

4択5問です。選んでから、答え合わせを開いてください。

Q1. 今日作った単一HTMLのアプリについて、情報システム部門への説明として最も通るのはどれか。

A. 企業向けの契約で使っているので安全だと伝える

B. 他社での導入事例を並べて、前例があることを示す

C. どこに何を送るか、どこに保存するか、誰が見られるか、やめるときどう消すか。この4点に具体名と数字で答える

D. リスクはゼロにできると断言して、審査を早く終わらせる

答え合わせ —

正解は C。審査の実務は「ツールが安全か」ではなく「誰の権限で、何に到達し、どこに残り、後から追えるか」に収束します。/secure が作る点検レポートは、この4点に答える形で書かれています。A と D は審査側の役割を否定する言い方なので、必ず止まります。限界を自分から先に出したほうが速く進みます。

Q2. 内製ツールの効果測定で、失敗を早く検知できる指標はどれか。

A. 作成から90日後に、週1回以上使われているツールの比率

B. 期間中に作成されたツールの累計本数

C. 1人あたりのAI利用量にもとづく社内ランキング

D. 研修を受講した人数と、その部署数

答え合わせ ー

正解は A。90日生存率です。作った数を数える組織は必ず失敗します。見るべきは生存側で、あわせて単一利用者率（作った本人しか使っていない比率）と最終更新からの経過日数を見ます。C は DORA が2026年6月2日の記事で虚栄指標として切り捨てたもので、個人ランキングは DX Core 4 も DORA も明示的に禁じています。

Q3. LINEヤフーの「3年で業務生産性2倍」という発表から、経営として読むべき箇所はどこか。

A. 2倍という倍率の大きさ

B. 全従業員約11,000人という対象規模

C. 全社にAIを配ったという意思決定の速さ

D. 対象が「従業員業務の3割」だと明示されている点

答え合わせ ー

正解は D。3割を対象にした2倍なので、全体では最大でも1.18倍です。この発表が誠実なのは分母を書いている点で、逆に言えば分母を書かない「生産性2倍」は読む価値がありません。自社で数字を作るときも、最初に決めるのは分子ではなく分母です。

Q4. 社内に広げたツールの「やめどき」を決める方法として、実際に機能するのはどれか。

A. 使い続ける人が現れなくなるまで、様子を見る

B. 作った時点で見直す時期を決めておき、その時点で使われていなければ捨てる

C. 情報システム部門から指摘が来た時点で止める

D. より良い後継ツールが出てから、まとめて入れ替える

答え合わせ ー

正解は B。作ったものは増え、増えたものは誰も面倒を見なくなります。RPAの野良ロボットで起きたことがそのまま起きます。作成時に見直し時期を決めておくだけで、1年後の棚卸しの重さが変わります。あわせて、異動したときの引き継ぎ先、誤りが出たときの責任者、保守工数の出どころの3つを決めます。3つとも埋まらないなら、個人の道具のままにしておくのが正解です。

Q5. 個人PCで、社内ネットワークにつながずにAIを使う運用形態の説明として正しいのはどれか。

A. 社内システムに到達しないので、統制上のリスクは考えなくてよい

B. 管理用の設定ファイルを配れば、会社側から統制できる

C. 統制が効かないことを明示したうえで、対象者・期間・扱ってよいデータ区分・終了後の消去手順を決めて、期限付きで受容する

D. 個人契約であれば会話が学習に使われないので、そのまま業務利用へ広げてよい

答え合わせ ー

正解は C。私物端末には管理設定を配布できないため、組織側の統制手段は実質ゼロになります。さらにセッションの会話は端末の `~/.claude/projects/` に平文で30日残り、その端末を会社は消去できません。シャドーITとの差は技術ではなく、承認記録と期限の有無だけです。B は管理下の端末でのみ成立します。D も誤りで、個人契約は学習許可がオンなら最大5年保持されます。

WRAP UP

持ち帰り

手元に残ったもの

4つ

- タスク管理アプリ（ `app.html` ）
- ご自身の業務アプリ（保管庫の `50_業務改善/` ）
- 安全点検レポート（ `点検レポート.md` ）
- 自部門の効果測定ボード

言えるようになったこと

1. **誰に、どの業務から使わせるか。**1部門で1件、業務プロセスに埋めるところから始めます
2. **作られたものを受け入れるかどうかを、何で判断するか。**デモで動いたことは基準になりません。完成の条件が判定できる形で書かれていて、それを1つずつ確かめたかどうかです
3. **情報システム部門に何と説明すれば通るか。**4点に具体名と数字で答え、限界を自分から先に出します
4. **3か月後に何が起きていたら成功で、何が起きていたら撤退か。**90日生存率と単一利用者率で判定します

コードの書き方は覚えなくて構いません。

次の一歩

1週間以内に、2本目を作る

持ち帰りはこの一点です。1本目は手順を覚えるのに時間がかかりますが、2本目は半分で終わります。題材は今日カタログで印を付けて、選ばなかったほうから取ってください。ゼロから考え直すと、それだけで着手が遅れます。

4か月後に効いてくる話

2026年12月9日、EU の新製品責任指令（Directive (EU) 2024/2853）が適用開始になります。ソフトウェアとAIシステムが明示的に「製品」と定義され、過失を立証しなくても責任を問える厳格責任がコード自体に及びます。「AIが書いた」は免責事由になりません。EU案件をお持ちのクライアントには直接効きます。

研修後に手元で引く資料は `05_補足/` に3本あります。指示の型、道具の作り方、機密情報の扱い。

HELP 困ったとき

症状	初動
セットアップファイルが開かない（Mac）	`Control` を押しながらクリックして「開く」を選びます。それでも開かなければ手を挙げてください
セットアップファイルが止められる（Windows）	SmartScreen の画面で「詳細情報」から「実行」を選びます
セットアップで「要対応」が出た	その画面のまま手を挙げてください。何が欠けているかが画面に出ています
`/pipeline` や `/secure` が候補に出ない	Claude Code の入力欄でスラッシュを1つ打つと候補が出ます。並んでいなければ、演習フォルダで起動し直します
`/build` が止まったように見える	3分は待ちます。5分以上動かないときは `Esc` を1回押して「いまどこまで進んでいますか。残りを続けてください」と送ります
エラーが出た	エラーの文をそのまま貼り付けます。1行だけ「原因を1行で教えてから直してください」と足すと精度が上がります
`app.html` を開いても真っ白	「app.html を開いても画面が真っ白です。ブラウザの開発者ツールに出ているエラーを確認する手順を教えてください」と送り、返ってきた手順どおりにエラーをコピーして貼ります
直したのに画面が変わらない	ブラウザで `Command` + `R`（Windows は `Ctrl` + `R`）を押して読み込み直します
入れたデータが消えた	別のブラウザで開いていないか確認します。データは、保存したブラウザにしか残りません
元に戻したい	`Esc` を2回押すと、どこまで戻すかを選べます。直近100個、30日ぶん。ターミナルのコマンドで消したファイルは戻りません
スクリーンショットが貼れない	Mac でも `Ctrl` + `V` です。`Cmd` + `V` は iTerm2 でのみ効きます。画像ファイルをウィンドウにドラッグしても入ります
日本語のファイル名が文字化けした	ZIP の展開時に起きます。「このフォルダの文字化けしたファイル名とフォルダ名を、すべて元に戻してください」と送れば直ります

Claude Code の画面が崩れる（Windows）	窓を最大化してから `claude` を起動し直します。Windows Terminal を使うと起きません
頼んでいない場所まで直された	制約が書けていません。`Esc` 2回で戻してから、触ってほしくない場所を明記して投げ直します

ヒントは `02_タスク管理アプリ演習/hints/` に6本あります。`step1_要件メモ.md`、`step2_pipeline.md`、`step3_build.md`、`step4_直す指示.md`、`step5_詰まったとき.md`、`step6_社内で使う.md`。手が止まったときだけ開いてください。先に開くと、考える機会がなくなります。