TRAINING

AI協働開発ハンズオン講座

情報系トラック Day2

既存Webアプリへの機能追加・モデル使い分け・テスト業務効率化

Givery

本資料の二次転載禁止について

本資料の二次転載を禁止します。

本資料は株式会社ギブリーが作成した著作物です。受講者ご本人の学習および業務での参照に限りご利用いただけます。社外への配布、SNS等への掲載、第三者への共有はご遠慮ください。

研修中の画面撮影・録画も、資料が写り込む範囲ではお控えください。

Day1からの接続

今日は、自分が書いていないコードに手を入れます

ここまでにやったこと

Day1は5行の仕様から worklog.html を作りました。1行壊して画面が真っ白になり、原因を推測せず症状だけを伝えて直したところまでやっています。仕様もコードも、自分が書いたものでした。

これからやること

今日の午前は逆になります。配布の作業記録アプリという他人が書いた index.html を読ませてから、機能を2つ足します。その前に70分かけて、チャットのモデルを切り替えたときに何が変わって何が変わらないかを見ます。

そのあとどこで効くか

午後は、今日打った前置きをファイル1枚に移してチームで共有する話と、テスト観点の抽出に進みます。Day3では今日足した機能に対してテストを書き、ブラウザの中で動かします。

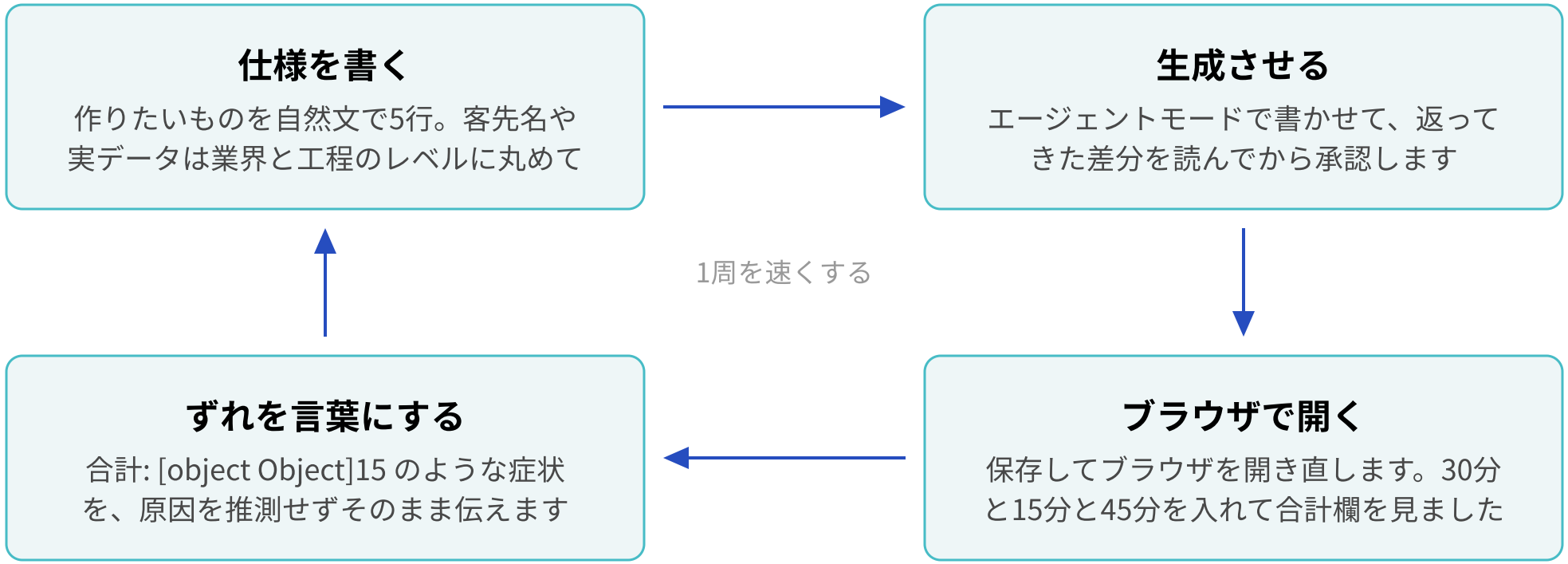
01

Day1の続きから

前回の4手を思い出し、今日1日の位置と手元の持ち物を確認します

Day1で回した4手

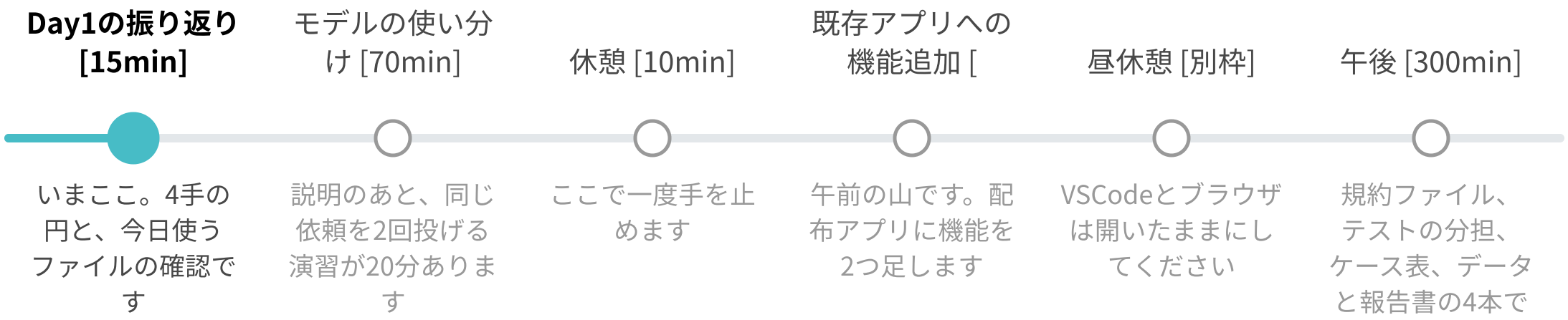
今日やることもDay3でやることも、全部この4手の中にあります



補足

今日1日の流れ

午前は177分。うち59分が各自で手を動かす時間です



案内は残り分数でします。戻る時刻は休憩に入るたびに口頭で伝えます。

手元にあるもの

tpro-work が見当たらない人は、いまこの場で作り直します

tpro-work フォルダ	デスクトップに置いたままのはずです。Day1に作ったファイルと、配布物がそのまま入っています。今日作るファイルも全部この中に入れます
worklog.html	Day1に自分で作ったアプリです。第2章の演習で、これを読ませる相手にします。無い人は配布の index.html で代用できます
index.html	配布の作業記録アプリです。日付・作業種別・内容・工数(時間) の4欄を持つ1ファイルで、第3章の題材になります
Copilotのサインイン	前回から1週間空いています。VSCode右下のアイコンに斜線が入っていたら、いま入り直してください

会社PCの運用でデスクトップが初期化される人がいます。無ければこの場で配布物を展開し直せば足ります。

モデルの話に入る前に

この70分は、どれが一番強いかの話ではありません

ここまでにやったこと

持ち物の確認が終わりました。tpro-work の中に、Day1に作った worklog.html と、配布の作業記録アプリの index.html が並んでいます。

これからやること

70分かけてモデルの使い分けを扱います。切り替え口の場所を全員で開き、同じ依頼を2回投げて違いを1行で書くところまでやります。順位の話はしません。

そのあとどこで効くか

休憩のあとの82分で、他人が書いたコードに機能を足します。うまくいかないとき最初に疑うのはモデルではなく渡した情報だ、という結論をこの70分で作ります。

02

モデルの使い分け

切り替え口の場所と、切り替えても変わらないことを自分の言葉で言えるようにします

選び方の前提

モデルは順位ではなく、いまの用途で選びます

> **順位を覚える話ではなく**
> **いまの用途で選ぶ話です**

Copilot Chat には複数の提供元のモデルが並びます。並ぶ顔ぶれは数か月で入れ替わるので、順位を覚えても次の現場では使えません。覚えるのは、切り替え口の場所と、切り替えても変わらないことの2つです。

解説

今日触る3つの製品

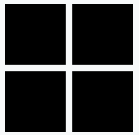
編集はVSCode、生成はGitHub Copilot、確認はブラウザです

VSCode

ファイルを開いて直す場所です。今日さわるindex.html もここで編集します

GitHub Copilot

VSCodeの中で動くチャットと補完です。モデルの切り替え口もこの中にあります



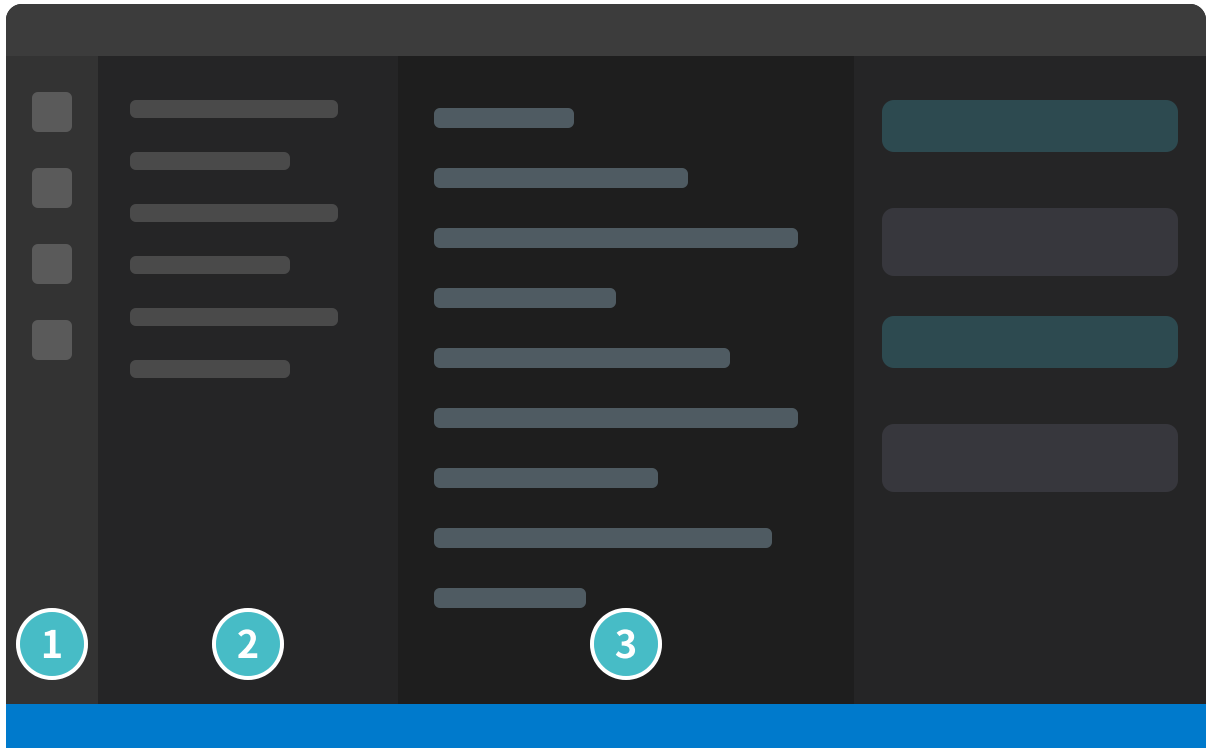
Edge

動かして確かめる場所です。保存したHTMLをここで開き直します

実演

切り替え口の場所

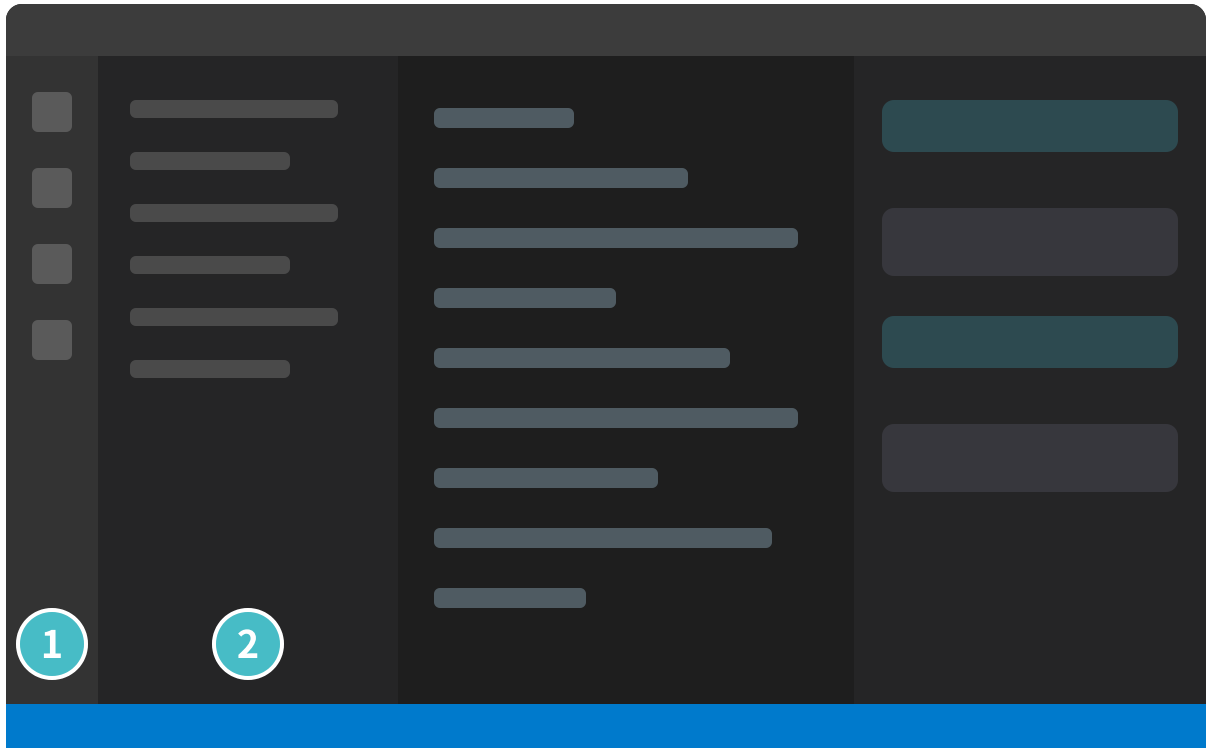
モデル名はチャット入力欄の中に出ています。そこが切り替え口です



1. **アクティビティバー**
   Copilotのアイコンを押すとチャットパネルが開きます。見当たらないときは Ctrl と Alt と I でも開きます。
2. **エクスプローラー**
   tpro-work の中の worklog.html をクリックして開いておきます。次の演習で読ませる相手です。
3. **エディタ**
   開いた worklog.html です。チャットに渡すファイルは、ここで開いているものから選ぶと早く済みます。

実演

切り替え口の場所（続き）



1 **チャットパネル**
入力欄の中に、いま使われているモデル名が出ています。クリックすると選択肢の一覧が開きます。ここが切り替え口です。

2 **ステータスバー**
右側にCopilotのアイコンが出ます。斜線が入っているときはサインインが切れています。

補足

並んでいる3系統

3つの提供元のモデルとAutoが、同じ場所に並んでいます

系統	提供元	よく使われる場面
Claude系	Anthropic	既存コードを読ませて説明させる。長いファイルをまとめて扱う
GPT・Codex系	OpenAI	小さな関数の追加。決まった形への書き換え
Gemini系	Google	画面や文章を含む依頼。下調べを兼ねた相談
Auto	Copilot側が選ぶ	どれを選ぶか決めきれないとき

> 並ぶモデルは入れ替わります。最新の一覧は公式ドキュメントで確認してください
> https://docs.github.com/en/copilot/reference/ai-models/supported-models

解説

切り替えの効き方

渡していない情報は、モデルを変えても補われません

切り替えで変わること

返答の言い回しと、コードの書き方の癖。長いファイルをどこまで一度に扱えるか。返ってくるまでの体感の速さ。この3つは切り替えると確かに変わります。

切り替えても変わらないこと

渡していない情報は、どのモデルも知りません。既存のコードを添付していなければ、どれを選んでも既存の作りを外した提案が返ってきます。ここが今日の午前の結論です。

解説

決め方は2つ

迷ったら既定のまま進め、外れたときだけ変えます

既定のまま使う	普段はこれで構いません。何も触っていない状態が既定です。作業の手を止めてまで選び直す必要はありません。
自分で選ぶ	同じ依頼で2回続けて期待から外れたときだけ、別の提供元の系統に切り替えて同じ依頼を投げ直します。依頼文は変えずに送るのがこつです。

依頼文とモデルを同時に変えると、どちらが効いたのか分からなくなります。変えるのは片方ずつです。

ワーク

演習 同じ依頼を2回投げる

同じ文を2回送ると、変わる部分と変わらない部分が自分の目で見えます

どんな場面か

派遣先では、使えるモデルが組織の設定で決まっていることがあります。選べる現場でも、切り替えて何がどう変わるのかを自分の目で見たことがないと、うまくいかないたびにモデルを変える癖がつきます。同じ文を2回投げて、変わる部分と変わらない部分を手元で確かめます。

やること

1. tpro-work の worklog.html をVSCodeで開きます。Day1のファイルが手元に無い人は、配布の index.html を使ってください。
2. tpro-work の中に model-memo.txt を新しく作り、依頼文を1行貼って控えます。文は「このファイルの動きを、初めて見る人に向けて説明してください。日本語で回答してください。」にそろえます。
3. チャット入力欄で半角の # を打って対象のファイルを添付し、モデル名は触らずにそのまま送ります。返ってきた説明は最初の5行だけ読みます。

ワーク

演習 同じ依頼を2回投げる（続き）

やること

1. 入力欄のモデル名をクリックして別の提供元の系統に切り替え、控えた文を貼ってもう一度送ります。文は1文字も変えません。
2. どこが違ったかを1行、model-memo.txt に書き足します。どちらが優れていたかは書きません。

できあがるもの

デスクトップの tpro-work フォルダの中に model-memo.txt。中に依頼文が1行と、違いを書いた1行が入っています。

ワーク

モデル比較の達成基準

違いが分からなかった、も記録として正しい結果です

達成の目安

- model-memo.txt に、送った依頼文がそのまま1行残っています。2回とも同じ文を送ったことが後から確認できます。
- 違いを書いた1行が入っており、優劣ではなく「どこが違ったか」の形で書かれています。

つまずいたら

- # を打っても候補が出ない。全角の＃になっています。半角に切り替えて打ち直します。
- 添付できていない。入力欄の上にファイル名のチップが出ているかを見ます。出ていなければ添付されていません。
- モデルの一覧が開かない。サインインが切れています。左下のアカウントアイコンからGitHubに入り直します。
- 一覧に系統が1つしか出ない。組織の設定で絞られています。その場合は既定のまま1回だけ送り、隣の人の画面で違いを見せてもらいます。
- 2回目で依頼文をつい足してしまった。控えた1行に戻して、新しいチャットで送り直します。

ワーク

モデル比較の達成基準（続き）

達成の目安

- 1回目は既定のまま、2回目は別の提供元の系統で送っています。提供元が変わっていることを一覧で確認できます。
- 違いが分からなかった場合は、そのまま「今回の依頼では差が分からなかった」と書けていれば達成です。

ワーク

モデル比較の達成基準（続き）

時間が余ったら

- 3つ目の系統でも同じ文を送り、3つを並べて違いを1行増やします。
- 同じモデルで2回続けて同じ文を送り、同じモデルの中でも返答が揺れるかを見ます。
- 依頼文の末尾に「200字以内で」を足して送り直し、字数の指定がどこまで効くかを見ます。

補足

比べるときの記録項目

良し悪しではなく、違いを言葉にして残します

見る点	書き方の例
説明の長さ	短くまとまった ／ 細かく並んだ
コードの出し方	全文を出した ／ 変更箇所だけ出した
前提の確認	先に質問してきた ／ そのまま答えた
自分の判断	この用途ならこちらを使う、と1行で書く

提出が発生する場合は言語化したテキストだけを出してください。コードや実データは提出できません。

クイズ

理解度チェック 1

Q. 配布された既存のWebアプリに、作業種別で絞り込む機能を足したい。最初にやることとして最も適切なものはどれですか。

- **A** 性能が高いとされる系統のモデルに切り替えてから依頼する
- **B** 対象のファイルをチャットに添付して、いまの作りを説明させる
- **C** 完成後の画面イメージを絵に描いてチャットに添付する
- **D** 「絞り込みをつけて」と送り、返ってきたコードをそのまま貼る

正解は次のページで確認します。まず自分で考えてみてください。

解説

解説

正解はB。先にいまの作りを渡すと、そのまま貼れる形の変更が返ります

B **対象のファイルをチャットに添付して、いまの作りを説明させる**

既存コードに手を入れる依頼は、いまの作りが渡っているかどうかで結果が変わります。先に読ませてから頼むと、既存の関数名とデータの持ち方をそのまま使った変更が返り、どこに足すかまで書かれます。切り替えは詰まってからで間に合います。

A 高性能とされる系統に切り替える
前提が渡っていない状態では、どの系統を選んでも既存の作りを外した提案になります。切り替えても渡していない情報は補われません。この章の結論がそのまま当てはまります。

C 完成イメージの絵を添付する
見た目の情報だけでは、既存の関数名やデータの持ち方が伝わりません。貼る場所が決まらず、結局は書き直しになります。

D いきなり依頼して結果を貼る
既存と噛み合わない書き直しが返り、貼ると一覧が二重に出るなどの手戻りになります。休憩明けの実演で、この状態を実際に見せます。

依頼文の最後に「まだコードは書かないでください」を添えると、説明だけを先に受け取れます。

配布アプリへの切り替え

ここから82分、題材は他人が書いた1ファイルになります

ここまでにやったこと

70分かけて、切り替えで変わることと変わらないことを整理しました。渡していない情報はどのモデルも知らない、というのが結論です。クイズで選んだのもその話でした。

これからやること

ここからの82分は、配布の作業記録アプリに機能を2つ足します。作業種別での絞り込みと、工数の合計です。読ませてから頼むという順序を、全員が自分の手で1回通します。

そのあとどこで効くか

午後は、この82分で毎回打つことになる前置きをファイル1枚に移します。今日足した2つの機能は、午後のテストケースと、Day3で書くテストの対象にもなります。

03

既存アプリへの機能追加

他人が書いた1ファイルを読ませてから、機能を2つ足せるようになります

解説

配布する作業記録アプリ

一覧と追加はありますが、絞り込みと集計はありません

項目	中身	今日の扱い
ファイル	index.html の1枚だけ。画面もスタイルも処理も全部この中に入っています	この1枚に追記していきます
入力欄	日付、作業種別、内容、工数(時間) の4つ。作業種別は 設計 / 実装 / レビュー / 試験 から選ぶ形です	絞り込みと合計の材料になります
データの持ち方	records という配列に date, kind, memo, hours の4項目で入ります。保存の仕組みは無く、再読み込みで消えます	項目名は変えません
まだ無い機能	作業種別での絞り込み、工数の合計、工数の入力チェックの3つ	今日の午前は上の2つを足します

追加した1件はブラウザを再読み込みすると消えます。保存の仕組みが無いアプリだと先に知っておいてください。

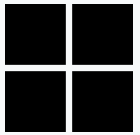
解説

書く場所と動かす場所

VSCodeで書き、ブラウザで開いて動かします

VSCode

index.html を書き換えて
Ctrl と S で保存します。
ここでは画面は動きません



Edge

保存した index.html を開
いて動かします。直したら
Ctrl と Shift と R で読
み直します

実演

順序で結果が変わる

同じ依頼でも、先に読ませたかどうかで返ってくるものが変わります

読ませずに頼んだとき

「作業種別で絞り込めるようにして」とだけ送ると、既存の render を使わない書き直しが返ります。貼ると一覧が二重に出たり、追加ボタンが反応しなくなったりします。

読ませてから頼んだとき

先に index.html を添付して説明させてから同じことを頼むと、既存の render と records をそのまま使った変更が返ります。どの関数の下に足すかまで書かれます。

解説

読ませる手順

読ませて返ってきた説明は、コードの中で自分で確かめます

1. **フォルダごと開く**
   VSCodeの「ファイル」から「フォルダーを開く」で tpro-work ごと開きます。ファイル単体で開くと、ほかのファイルを参照できません。
2. **いまの動きを先に見る**
   index.html をブラウザで開き、一覧が出ることと1件追加できることを確認します。壊す前の状態を自分の目で見ておきます。
3. **チャットにファイルを渡す**
   Copilot Chat の入力欄で半角の # を打ち、index.html を選びます。入力欄の上に添付したファイル名が並びます。
4. **説明だけを頼む**
   次のページの依頼文をそのまま送ります。ここではまだコードを書かせません。
5. **答えを自分で確かめる**
   返ってきた関数名を、実際に index.html の中で検索して探します。見つからなければ、その説明は当てになりません。

実演

読ませるための依頼文

最後の一行が、先回りしたコード生成を止めます

そのまま貼って使える依頼文です。index.html を添付してから送ってください。3点の中身を書き換えれば、ほかのアプリでもそのまま使えます。

```
#index.html
このアプリの役割と、画面に一覧が出るまでの流れを説明してください。
次の3点は必ず含めてください。
- 作業記録のデータがどこに入っているか
- 一覧を描画している関数の名前
- 追加ボタンを押したときに呼ばれる関数の名前
日本語で回答してください。まだコードは書かないでください。
```

> 3点を指定しているのは、返ってきた説明を自分で検算できる形にするためです。関数名は検索すれば当たりを確かめられます。

ワーク

演習 作業種別での絞り込み

仕様を3行書いてから頼むと、貼る場所まで返ってきます

どんな場面か

派遣先で最初に任される改修は、たいてい他人が書いた画面への小さな追加です。仕様書は無く、動く画面と口頭の依頼だけが手元にあります。その状態から、既存を壊さずに1機能を足すところまでを、配布の作業記録アプリで一度通します。

やること

1. 仕様を3行でチャットに書きます。一覧の上に作業種別を選ぶ欄を置くこと、選ぶと一覧がその作業種別だけになること、「すべて」を選ぶと元に戻ること。この3行です。
2. 半角の # で index.html を添付し、3行の仕様を送ります。末尾に「既存の render をそのまま使ってください」と一文足します。
3. 返答に書かれた場所を先にVSCodeで開いて見つけてから貼ります。分からなければ「どの関数の下に入れるか教えてください」と聞き返します。

1回に1つだけ変えてください。2つ同時に頼むと、どちらが原因で壊れたか分からなくなります。

ワーク

演習 作業種別での絞り込み（続き）

やること

1. Ctrl と S で保存し、ブラウザで Ctrl と Shift と R を押して読み直します。選択を切り替えて一覧が変わるかを見ます。
2. 動かなければ F12 を押し、出ているエラー文をそのままチャットに貼ります。自分の言葉に言い換えないでください。

できあがるもの

デスクトップの tpro-work フォルダの中の index.html に、作業種別を選ぶ欄が追加され、選択で一覧が切り替わる状態になります。

ワーク

絞り込みの達成基準

足した機能が動くことと、前からある機能が壊れていないことは別に見ます

達成の目安

- 一覧の上に作業種別を選ぶ欄が出ており、設計を選ぶと設計の行だけが残ります。
- 「すべて」に戻すと、絞り込む前の件数に戻ります。

つまずいたら

- 選択欄は出たが一覧が変わらない。イベントの登録が漏れています。「選択を変えたときに一覧が更新されません」とだけ伝えて直させます。
- 画面が真っ白になった。F12 を押して赤い行の1行目をそのまま貼ります。原因は推測しません。
- 保存したのに画面が変わらない。ブラウザのキャッシュです。Ctrl と Shift と R で読み直します。
- 追加ボタンが反応しなくなった。既存の render が置き換わっています。貼る前に戻して、既存の render を使うよう書き足して頼み直します。
- 貼る場所が分からない。「どの関数の下に入れるか教えてください」と聞き返せば、行の位置まで返ってきます。

ワーク

絞り込みの達成基準（続き）

達成の目安

- 絞り込んだ状態でも、フォームから1件追加できます。前からある機能が壊れていません。
- index.html の中に、既存の render を呼ぶ形で追記されています。render が丸ごと別の関数に置き換わっていません。

ワーク

絞り込みの達成基準（続き）

時間が余ったら

- 選択中の件数を、一覧の上に「表示 12 件」のように出します。
- 作業種別に加えて、日付の範囲でも絞り込めるようにします。ただし1つずつ足してください。
- 絞り込んだ状態で追加したとき、その1件が一覧に出るかどうかを確かめ、出ない場合の仕様を自分で決めます。

ワーク

演習 作業種別ごとの工数合計

AIが出した合計は、1つだけ手で検算します

どんな場面か

集計はAIがすぐ書いてくれる部類の処理です。だから中身を見ずに通してしまいやすく、桁が違っていても画面はそれらしく見えます。出てきた数字を人が1つだけ検算する動作を、ここで手癖にします。

やること

1. 何を出すか決めます。作業種別ごとの工数の合計を、一覧の下に表で出します。絞り込みが効いているときは、その結果だけを合計します。
2. 同じように index.html を添付して仕様を送ります。末尾に「前の絞り込み機能を壊さないでください」と一文足します。
3. 作業種別を1つ選び、一覧に出ている工数を自分で足して、画面の合計と一致するかを見ます。ここは必ず人がやります。

ワーク

演習 作業種別ごとの工数合計（続き）

やること

1. 作業種別を切り替えて、合計も一緒に変わるかを確認します。変わらなければ、その症状をそのまま伝えて直させます。
2. 検算の結果を1行、tpro-work の中の model-memo.txt に書き足します。合っていたか、ずれていたか、ずれた原因は何だったかを書きます。

できあがるもの

デスクトップの tpro-work フォルダの中の index.html に合計の表が追加され、model-memo.txt に検算の結果が1行増えます。

ワーク

合計機能の達成基準

画面に数字が出たことではなく、その数字が合っていることを見ます

達成の目安

- 一覧の下に、作業種別ごとの工数合計が表で出ています。
- 作業種別を1つ選んで手で足した数と、画面の合計が一致しています。一致を自分で確認しました。
- 絞り込みを切り替えると、合計も一緒に変わります。
- model-memo.txt に検算の結果が1行増えています。ずれていた場合は、ずれた原因も書かれています。

つまずいたら

- 合計が全件のままで、絞り込みと連動しない。「作業種別を切り替えても合計が変わりません」とだけ伝えます。
- 合計が 1.52.5 のように横につながる。文字列として足されています。症状をそのまま貼れば直ります。
- 合計の表が一覧の中に入り込んで崩れる。どこに出すかを仕様に書いていません。「一覧の下に別の表として出してください」と足します。
- 前の絞り込みが消えた。手順2の一文を書き忘れています。貼る前の形に戻してから頼み直します。

ワーク

合計機能の達成基準（続き）

時間が余ったら

- 合計の多い順に並べ替えます。
- 合計行に件数を併記し、何件で何時間かが分かるようにします。
- 合計を1つだけわざと壊してもらい、検算で気づけるかを試します。壊す前のファイルは必ずコピーしてから行います。

発展

壊れたときの戻り方

動いた形を残しておくと、いつでもそこへ戻れます

動いた形を残す

機能が1つ動いたら、index.html をコピーして別名で保存します。
ファイル名の末尾に _ok1 のように付けます。

→

1回に1つだけ変える

2つ同時に頼むと、どちらが原因で壊れたか分かりません。頼む前に、何を1つ変えるかを決めます。

→

戻して頼み直す

手に負えなくなったら、残しておいた形に戻します。そこから依頼文を書き直して、もう一度頼みます。同じ文で投げ直しても同じところで詰まります。

> 本研修ではGitを使いません。コピーで代用しています。実務でGitが使える環境なら、そちらのほうが確実です。

午前の結論

うまくいかないとき、モデルより先に渡した情報を疑います

> **読ませてから頼む**
> **モデルより先に、渡した情報を疑う**

午前に足した2つの機能は、午後のテストケースの題材になります。工数の入力チェックは今日は足しません。まだ無い機能のケース表を午後に作り、Day3で実装して動かします。

午前からの引き継ぎ

うまく頼めた依頼文は、いま**その人のチャットの中にしか**ありません

ここまでにやったこと

午前は配布の作業記録アプリを Copilot に読ませてから頼み、作業種別での絞り込みと工数の合計を足しました。講師の画面では、読ませずに頼むと既存の関数を無視した書き直しが返ることも見ました。

これからやること

毎回の前置きとして打っていた指定を、.github/copilot-instructions.md というファイル1枚に移します。まず全文を見て、そのあと自分の tpro-work に置き、同じ依頼を前後で1回ずつ出して差を見ます。

そのあとどこで効くか

午後の後半でテストケースを大量に出させるとき、この規約ファイルが出力の形をそろえます。Day3 でテストコードを書かせるときも、置き場所と書き方は今日と同じです。

04

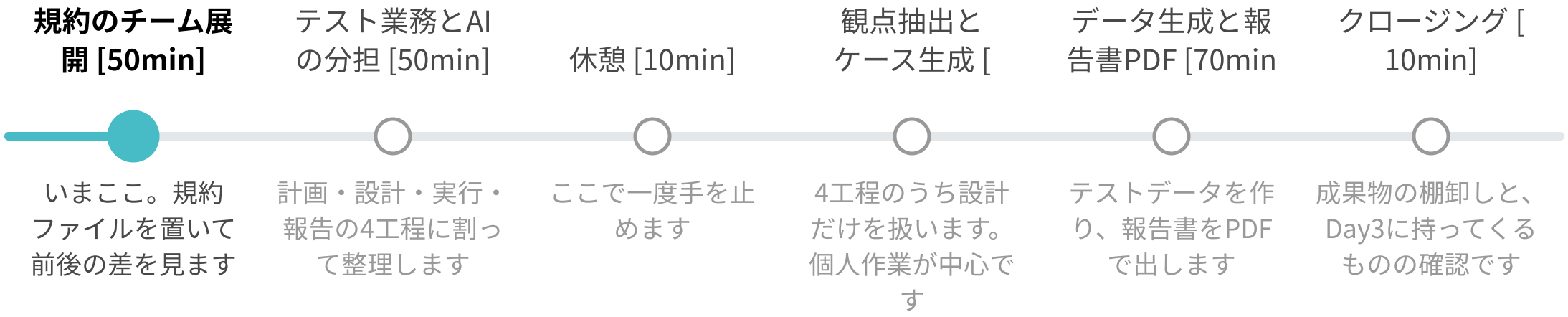
規約のチーム展開

毎回の前置きをファイル1枚に移し、置き場所と中身を自分で書ける状態にします

補足

午後の進み方

午後300分のうち、各自で手を動かすのは92分です



休憩は午後の真ん中に1回入ります。それまでの100分は続けて進みます。

解説

前置きの置き場所

毎回打っていた前置きは、ファイル1枚に移して省けます

チャットの冒頭に毎回打つ

新しいスレッドを開くと前の会話は引き継がれないので、使う言語、禁止したいライブラリ、コメントの書き方を打ち直します。打つ内容は人によって違うので、同じ画面のコードでも出てくる形がそろいません。前置きを省いた人のコードだけ、レビューで指摘が増えます。

決まった場所のファイルに書く

tpro-work/.github/copilot-instructions.md に置くと、Copilot Chat が会話のたびに自動で読みます。受講者が意識して貼る必要はありません。書き方を変えたいときは、このファイル1か所を直せば全員の依頼に反映されます。

解説

規約ファイルの全文

禁止したいことまで書くのが要点です

作業記録アプリ向けに書いた copilot-instructions.md の全文です。前提・言語と書き方・レビュー観点・変更の出し方の4つの見出しで区切っています。演習ではこの内容をそのまま写して構いません。

```
# このプロジェクトの前提
- 単一の index.html に HTML と CSS と JavaScript を書く。ファイルを分割しない。
- ブラウザで index.html を直接開いて動かす。ビルド手順は無い。
- fetch は使わない。file:// では動かないため、ファイルはドラッグ＆ドロップで受け取る。

# 使う言語と書き方
- JavaScript（ES2020）で書く。TypeScript は使わない。
- 外部ライブラリを追加しない。同梱しているものだけ使う。
- 変数名と関数名は英語、コメントは日本語で書く。
- DOM の取得は document.getElementById と querySelector に統一する。
```

解説

規約ファイルの全文（続き）

前ページの続きです。

```
- 既存の render 関数は消さない。画面の描き直しは render を呼んで行う。

# レビューで見るところ
- 入力値のチェックを通ったあとに records へ追加しているか。
- 画面に出す文言をコードの各所に散らさず、1か所にまとめているか。
- 例外を握りつぶさず、画面のメッセージ領域に文言を出しているか。

# 変更の出し方
- 変えたファイル名と関数名を先に1行で書く。
- 該当する関数は省略記号を使わず全体を出す。
```

補足

GitHubとCopilotの関係

.github はGitHub由来のフォルダ名で、読むのはCopilotです

GitHub

ソースコードを共有する場所です。.github はその設定を置く決まったフォルダ名です

GitHub Copilot

VSCodeの中で動く別の製品です。会話のたびに .github の中の規約ファイルを読みます

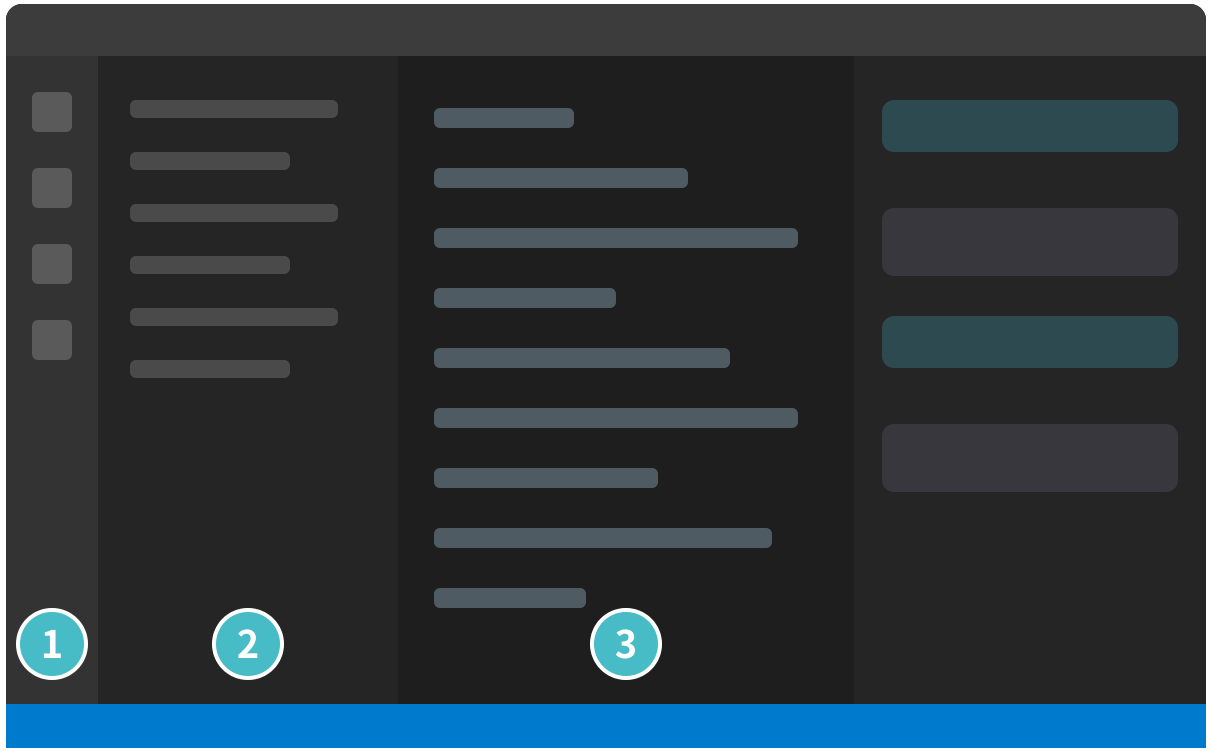
VSCode

フォルダを開く場所です。.github もエクスプローラーからここで作ります

実演

規約ファイルの置き方

tpro-work の直下に .github を作ります。1つ上に作ると読まれません



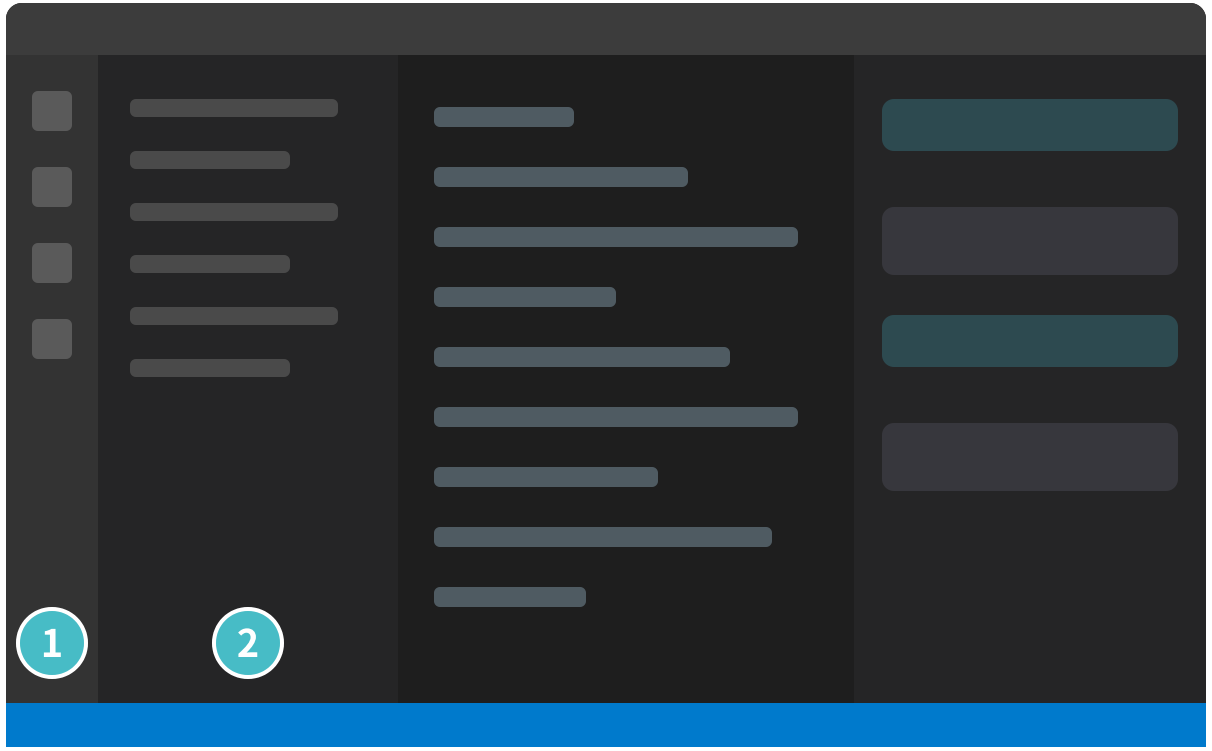
1. **アクティビティバー**
   一番上のファイルのアイコンを押すと
   エクスプローラーが開きます。閉じているときはこ
   こから戻ります。

2. **エクスプローラー**
   tpro-work の行を1回クリックして選び、新しい
   フォルダーのアイコンで .github を作ります。先
   頭のドットを忘れると読まれません。

3. **エディタ**
   .github を選んだ状態で新しいファイル
   copilot-instructions.md を作り、前ページの全文
   を貼って Ctrl+S で保存します。

実演

規約ファイルの置き方（続き）



1 チャットパネル
依頼を送ったあと、回答の上に出る参照ファイルの一覧に copilot-instructions.md が並んでいるかを見ます。

2 ステータスバー
タブ名の横の丸印が消えていれば保存済みです。丸が付いたままのファイルは読まれません。

解説

2つの規約ファイル

読む側が違うので、置き場所と書く粒度も変わります

項目	copilot-instructions.md	AGENTS.md
置き場所	tpro-work/.github/ の中	tpro-work の直下
読む側	GitHub Copilot。Chat が会話のたびに読みます	Copilot を含む複数のAI コーディングツールが読む共通の書式です
入れる中身	書き方の指定。言語、命名、禁止事項、レビュー観点	プロジェクトの前提。構成、動かし方、触ってよい範囲
分量の目安	20行前後。守ってほしい順に上から	30行前後。事実だけを書き、判断は書かない

> 同じ内容を両方に写さないでください。片方を直したときにもう片方が古くなり、結果が安定しなくなります。

補足

AGENTS.mdの全文

AGENTS.md には判断ではなく事実だけを書きます

同じ作業記録アプリ向けの AGENTS.md です。構成と動かし方、触ってよい範囲を書いています。tpro-work の直下に置きます。

```
# 作業記録アプリ
日付・作業種別・内容・工数を入力して一覧に表示する、1画面のアプリです。

## 構成
- index.html … 画面、スタイル、処理を1ファイルに収めています
- .github/copilot-instructions.md … コーディング規約

## 動かし方
- index.html をブラウザで開きます。サーバは立てません。
```

補足

AGENTS.mdの全文（続き）

前ページの続きです。

```
- 直したら Ctrl+S で保存し、ブラウザで Ctrl+Shift+R を押して読み直します。

## 触ってよい範囲
- records に入れる項目（date, kind, memo, hours）の名前は変えない。
- 既存の render を消して作り直さない。追記で足す。

## 制約
- 外部と通信する処理は書かない。
- 実在する会社名・人名・案件名をデータに書かない。
```

ワーク

演習 規約ファイルの設置

同じ依頼を規約ファイルの前後で1回ずつ出し、差を目で見ます

どんな場面か

派遣先では、同じ画面のコードを複数人が触ります。前置きを丁寧に打つ人と省く人がいると、同じ依頼でも返ってくるコードの形がそろわず、レビューの指摘が人によって変わります。規約ファイルを1枚置いたとき、その差がどこまで埋まるのかを自分の手元で確かめます。

やること

1. デスクトップの tpro-work フォルダをVSCodeで開き、Copilot Chat に「工数の入力チェックの関数を追加して」とだけ送ります。言い回しは変えません。
2. 出てきたコードを tpro-work フォルダの中の before.txt に貼って保存します。
3. エクスプローラーで tpro-work を選び、新しいフォルダーで .github を作り、その中に copilot-instructions.md を作ります。

ワーク

演習 規約ファイルの設置（続き）

やること

1. 「規約ファイルの全文」のページの全文を貼り、自分の現場で足したい行を2行だけ追記して Ctrl+S で保存します。
2. チャットを新しいスレッドで開き直し、手順1とまったく同じ文言をもう一度送ります。出てきたコードを after.txt に貼って保存します。
3. before.txt と after.txt を左右に並べて開き（タブを右端にドラッグすると画面が2分割になります）、変数名・コメントの言語・DOMの取得方法の3点を見比べます。

できあがるもの

デスクトップの tpro-work フォルダの中に、before.txt と after.txt、それに .github/copilot-instructions.md の3つが残ります。

ワーク

演習の達成基準

読まれたかの確認と、守られたかの確認は別の作業です

達成の目安

- 回答の上に出る参照ファイルの一覧に copilot-instructions.md が並んでいます。ここまでで「読まれた」ことが確認できます。
- 規約に書いた項目のうち、何個が反映されて何個が反映されなかったかを数で言えます。

つまずいたら

- .github フォルダが作れない。Windows のエクスプローラーはドットで始まる名前を弾くことがあります。VSCode のエクスプローラーで新しいフォルダーを押し、名前に .github と入力します。
- 参照ファイルの一覧に名前が出ない。保存していないか、tpro-work の1つ上に .github を作っています。タブの丸印と、エクスプローラー上の階層の2つを確認します。
- 前後で差が出ない。手順1の依頼文に細かい指定を足した人は、規約と内容が重なって差が消えます。文言を元に戻してもう一度送ります。
- 2回目を同じスレッドで続けてしまった。前の会話に引きずられて差が見えません。新しいスレッドを開いて送り直します。
- ファイル名を copilot-instruction.md と書いた。instructions は複数形です。1文字違うと読まれません。

ワーク

演習の達成基準（続き）

達成の目安

- after.txt のコメントが日本語で、DOM の取得が getElementById か querySelector になっています。
- before.txt と after.txt が tpro-work フォルダの中に保存されています。

ワーク

演習の達成基準（続き）

時間が余ったら

- 規約に「1関数30行以内」を追記して3回目を送り、行数の指定がどこまで効くかを見ます。
- 禁止事項の行だけを消した版で同じ依頼を送り、否定形があるときとないときで結果が変わるかを見ます。
- AGENTS.md を tpro-work の直下に置き、参照ファイルの一覧に2つとも出るかを見ます。

テストへの切り替え

ここから Day2 の残りと Day3 は、全部テストの話です

ここまでにやったこと

規約ファイルを tpro-work の直下に置き、同じ依頼から返ってくるコードの形がそろうことを確認しました。効かない指定があることも見えました。

これからやること

話題をテストに移します。
テスト業務を計画・設計・実行・報告の4工程に割り、AIに候補を出させる場所と、人が決める場所を工程ごとに対で見ます。
この50分は手を動かしません。

そのあとどこで効くか

休憩のあとの110分は、4工程のうち設計だけを扱います。いま何の作業をしているのか分からなくなったら、この4工程の図に戻ります。Day3 の実技課題も同じ4工程の中にあります。

05

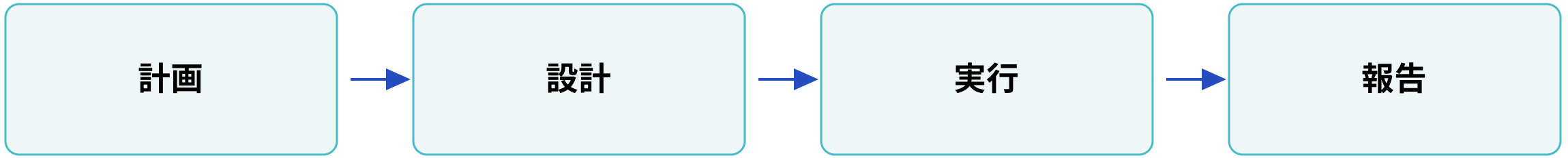
テスト業務とAIの分担

4工程のどこでAIに何を出させ、何を自分で決めるかを言えるようにします

解説

テストの4工程

今日このあと触るのは、4工程のうち**設計だけ**です



何をどこまで確認するかを決めます。範囲と、ここまでやれば終わりという条件を書く工程です。

観点を並べ、条件を組み合わせ、ケース表にします。休憩のあとの110分はここだけをやります。

手順どおりに動かし、画面を目で見て結果を記録します。動かすのはDay3 です。

件数と未実施を数え、読む人に合わせて要約します。今日の最後にPDFで出します。

> この4工程は、単体テストでも結合テストでも同じ順に並びます。工程の名前だけ覚えて帰ってください。

解説

AIが出せる候補

どの工程でもAIが出すのは候補までです

計画	過去に起きた不具合の傾向から、確認したい範囲の候補を並べます。抜けている観点を指摘させる使い方もできます。
設計	観点の洗い出し、条件の組み合わせ、ケース表の下書き。30件の下書きを2分で出します。
実行	操作手順の文章化、結果の整形、再現手順の書き起こし。動かした人のメモを読める文にします。
報告	件数の集計、表の整形、文章の体裁。優先度別の内訳を数えて表にします。

ここに挙げたものは全部、間違っていても画面上は同じ見た目で返ってきます。

解説

人が決めること

前ページと同じ4工程です。決める側は動きません

計画	どこまでやれば出荷してよいかの線引き。範囲を狭める判断と、その理由を残すこと。
設計	仕様の解釈、優先度、落としてよいケースの判断。30件の下書きから何を捨てるかを決めます。
実行	実際に動かして画面を目で見る確認と、環境が妥当かの判断。ここは今日もDay3も人がやります。
報告	不具合の重大度、リリース可否、誰にどう伝えるか。業務への影響を知っている人しか決められません。

前ページとこのページを並べると、どの工程も「候補を出す」と「決める」の2つに割れています。

補足

責任の置き場所

自分の名前で判子を押せるかが、そのまま任せてよい範囲の線になります

> **「テストの網羅性を保証したのはAIです」**
> **と書ける報告書は存在しません**

ケースが足りていませんでした、と説明する場面で、AIが出したので、は通りません。出したものに自分の名前を出せる範囲までが、AIに任せてよい範囲です。判断を任せた分だけ、説明できない部分が増えます。

実演

返る量を止める1行

書かせないものを1行で禁止すると、返る量が制御できます

一度に100件のケースを出させると、1件ずつ見る気力がなくなります。分けて出させる最初の1手は、書かせないものを1行で禁止することです。効いているのは2行目だけです。

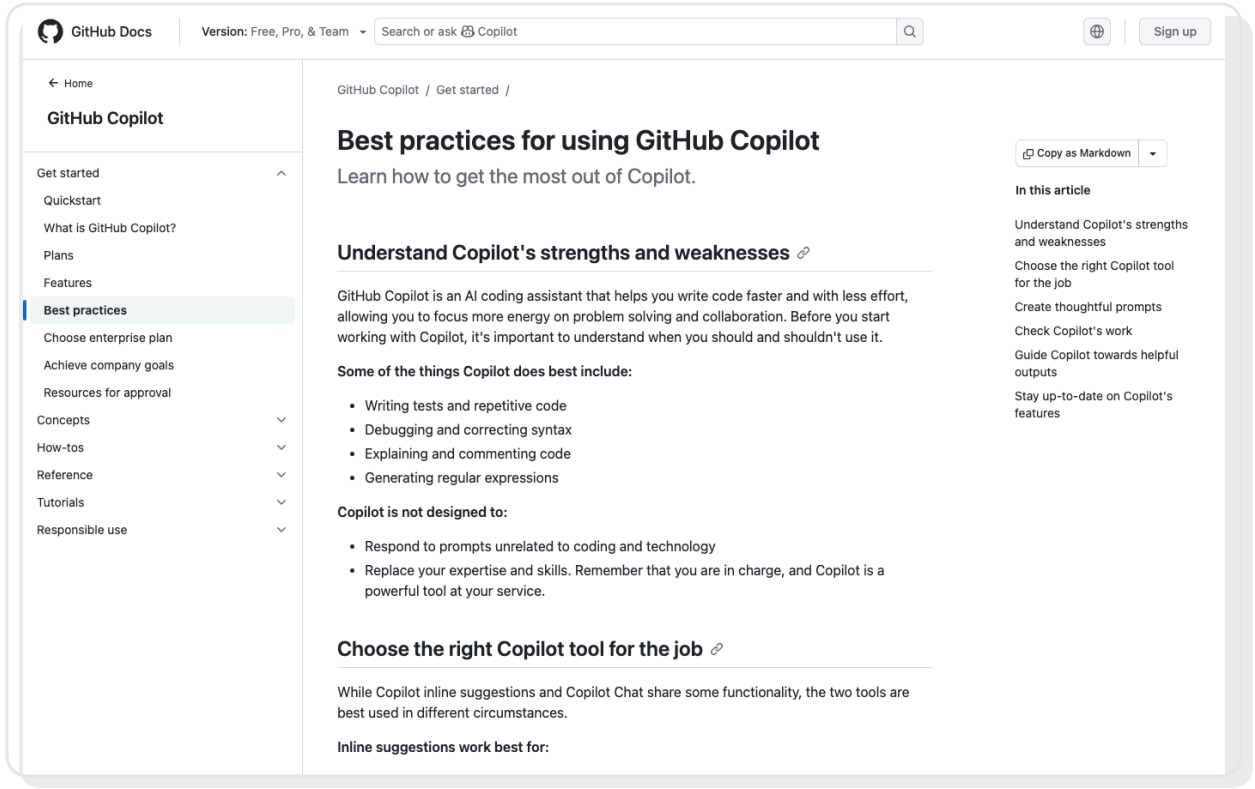
作業記録アプリの工数の入力チェックについて、テスト観点を出してください。
テストケース、期待結果、操作手順は書かないでください。

この2行に対象の絞り込みと個数と書式を足した完成形は、休憩明けの章で渡します。

補足

公式が挙げる注意点

レビューせずに使わないこと、依頼を小さく分けること



GitHub 公式のベストプラクティスのページ。生成結果をレビューせずに使わないこと、依頼を小さく分けることが挙げられています。記載は更新されるので、社内に展開する前にこのページで現在の内容を確認してください。

クイズ

理解度チェック 2

Q. テスト業務でAIに任せてよい範囲の線引きとして、最も適切なものはどれですか。

A　観点の洗い出しは人がやり、ケース表の清書だけをAIに任せる

B　過去の不具合の傾向を渡し、どこまで確認すれば出荷してよいかの判断まで任せる

C　候補を出す作業はAIに任せ、どれを落とすかと不具合の重大度は人が決める

D　100件のケースを一度に出させ、レビューは不具合が出た箇所だけに絞る

正解は次のページで確認します。まず自分で考えてみてください。

解説

解説

正解はC。どの工程でも、候補はAI、決めるのは人という同じ形になります

C

候補を出す作業はAIに任せ、どれを落とすかと不具合の重大度は人が決める

計画・設計・実行・報告の4工程のどこを見ても同じ形です。AIは候補を量産できますが、落としてよいケースの判断と不具合の重大度は、その現場と業務への影響を知っている人にしか決められません。報告書に自分の名前で判子を押せる範囲が、そのまま任せてよい範囲の線になります。

A

観点の洗い出しは人がやり、清書だけをAIに任せる

観点の洗い出しはAIが速い作業で、人の手を残す価値が薄いところです。人に残すべきなのは清書ではなく、並んだ観点から何を落とすかの判断です。分担が逆になっています。

B

出荷してよいかの判断まで任せる

傾向から範囲の候補を出させるところまでは使えます。ただし終了条件の判断は人に残ります。そのまま出すと、根拠のない工数見積もりや、狭すぎる範囲がそのまま計画になります。

D

100件を一度に出させ、レビューは不具合が出た箇所だけ

一度に大量に出させるとレビューが形だけになります。公式も依頼を小さく分けることを挙げています。不具合が出た箇所だけを見る運用では、そもそもケースが抜けていることに気づけません。

休憩のあとは、この線引きのうち設計の工程だけを110分かけて手で通します。

設計工程への入り口

観点とケース表づくりが、今日いちばん長い110分です

ここまでにやったこと

規約ファイルで頼み方の下地を作り、テスト業務のどこにAIが効くかを計画・設計・実行・報告の4工程で見ました。ここまでが午後の前半100分です。

これからやること

ここからは設計工程だけを110分で深掘りします。観点を出し、条件に分け、ケース表にして、抜けを足す。うち48分は各自で手を動かす時間です。

そのあとどこで効くか

できたケース表は、このあとの70分でテストデータと報告書の材料になります。Day3では同じ表を単体テストとE2Eに落として、実際に動かします。

今日は実行しません。実行はDay3です。今日の成果物はケース表とデータと報告書の3つです。

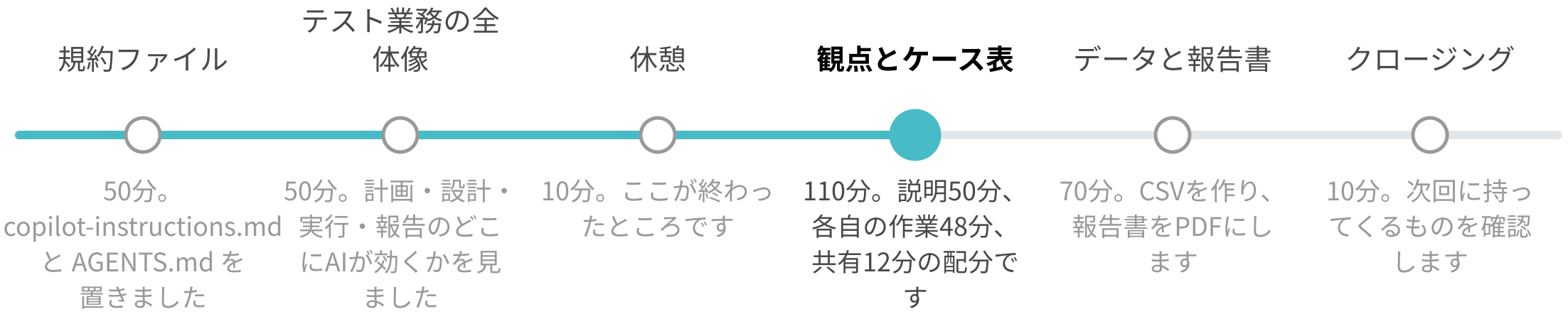
06

観点抽出とテストケース生成

観点から期待結果までを4段に分けて落とし、抜けを足す改善サイクルを2周回せるようになります

午後の残り190分

いまは午後の休憩明けです。ここから残りは190分です



昼休憩は午前と午後の間に入っています。午後は途中に10分の休憩が1回で、いまその直後です。

解説

4段の構造

観点からケース表までを4段に分けて、順に細かくします

段1 観点

何を確かめたいかの切り口です。工数の入力チェック、作業種別の絞り込み、作業種別ごとの工数合計、という粒度で並べます

→

段2 条件

観点ごとに値の範囲や状態に分解します。工数なら 0以下、0.5から24、24超、空欄、全角数字の5つです

→

段3 ケース

条件を1行の操作に落とします。工数に 0 を入れて追加ボタンを押す、という書き方です

→

段4 期待結果

そのとき画面がどうなるかを書きます。出るメッセージの文言と、一覧が増えるかどうかまで書きます

> 段を飛ばして一気に頼むと、画面全体の話とボタン1つの話が混ざった表が返ってきます。

解説

配布アプリでの記入例

配布の作業記録アプリの工数で、段1から段4まで1本通します

段1 観点	工数の入力チェック
段2 条件	0以下 ／ 0.5から24 ／ 24超 ／ 空欄 ／ 全角数字 の5つに分ける
段3 ケース	工数に 0 を入れて追加ボタンを押す
段4 期待結果	工数欄の下に「0.5以上24以下で入力してください」が出る。一覧は増えない

対象は画面全体にしません。入力欄1つ、または選択欄1つまで絞ります。

解説

観点の出どころ

観点は仕様と画面の両方から出すと、片寄りが減ります

仕様だけから出す

仕様のテキストをチャットに貼り、テスト観点を箇条書きで出させます。速く出ます。ただし仕様に書かれていない入力欄や、押せてしまうボタンは出てきません。書き漏れがそのまま抜けになります。

仕様と画面の両方から出す

仕様のメモを貼ったあと、動いている画面のHTMLもそのまま貼り、この画面で操作できる場所を全部挙げてと頼みます。仕様に載っていない操作がここで出ます。重複は後でまとめれば済みます。

実演

観点を出させる文

表にしないでと書かないと、いきなりケース表が返ってきます

休憩前に見せた禁止の1行に、対象と個数と書式を足した完成形です。対象を1つに限り、個数の上限を付けます。この2つが無いと、画面全体の観点が30個返ってきます。

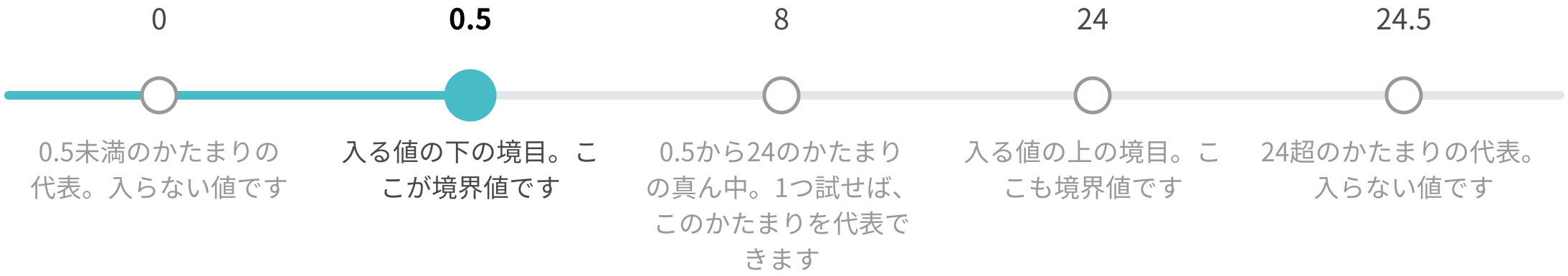
```
次のWebアプリの一部について、テスト観点だけを箇条書きで出してください。
- 対象は工数の入力欄と追加ボタンだけです
- 表にせず、観点の名前だけを1行ずつ書いてください
- 10個以内にしてください
- 観点の名前は「◯◯の入力チェック」のような短い名詞で書いてください

（この下に、対象部分のHTMLを貼ります）
```

> 出てきた観点をそのまま使いません。自分の観点を2つ足してから次の段に進みます。

解説

同値クラスと境界値

同値クラスは同じ結果になる値のかたまり、境界値はその境目です



線に載らない値も、同値クラスの1つとして拾います。空欄、全角数字、マイナスの3つです。

解説

条件の分け方

項目ごとに分けて書くと、ケースの数が先に読めます

入力項目	同値クラス	境界値として拾う値
工数（0.5から24の時間）	範囲内 ／ 範囲より小さい ／ 範囲より大きい ／ 数値でない	0、0.5、24、24.5
内容（上限は仕様で決める）	文字が入っている ／ 空 ／ 上限を超えている	空、1文字、上限、上限＋1文字
日付	当日以前 ／ 未来日 ／ 書式が違う	前日、当日、翌日
作業種別の選択欄	特定の工程を選択 ／ すべて	すべてに戻したとき

> 全部の組み合わせは作りません。1項目ずつ動かし、関係しそうな項目だけ組み合わせます。

補足

ケース表の列

列を先に指定しないと、AIは毎回違う形で返してきます

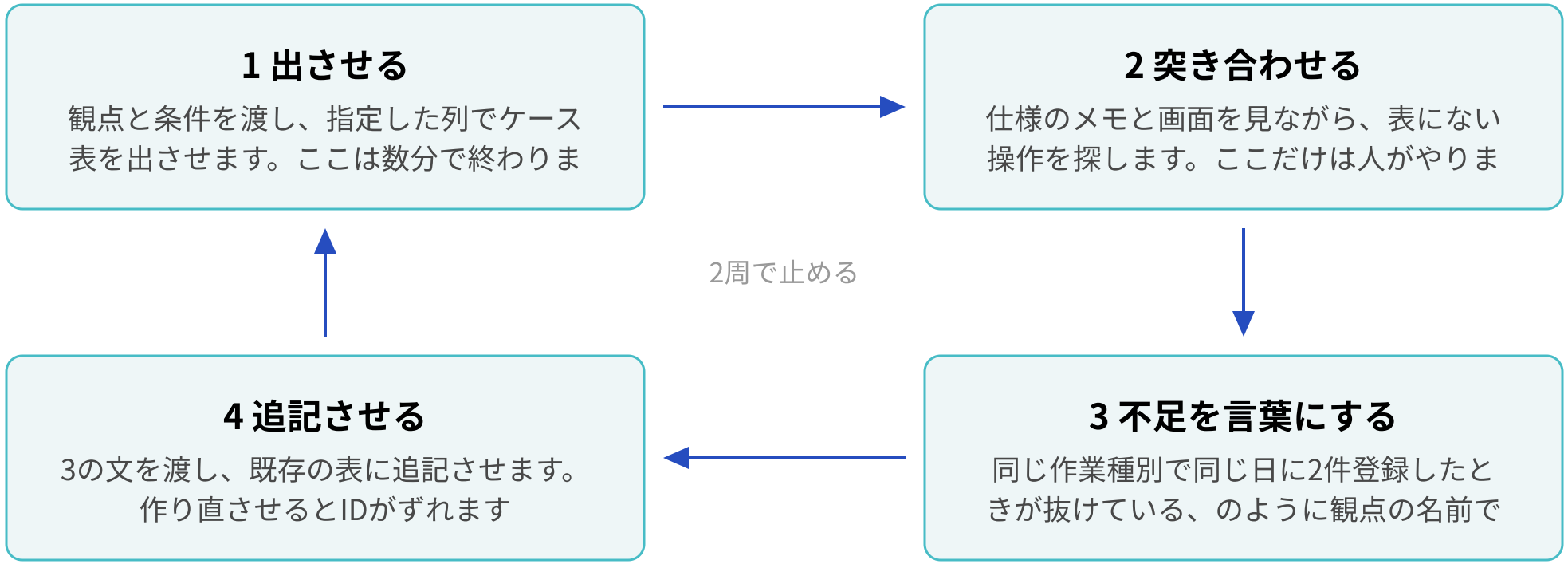
この見出し行をそのままコピーしてチャットに渡します。演習でも同じ行を使います。

```
| ID | 観点 | 前提 | 操作 | 期待結果 | 優先度 |
|---|---|---|---|---|---|
| C-01 | 工数の入力チェック | 一覧に初期データが表示されている | 工数に 0 を入れて追加を押す | 「0.5以上24以下で入力してください」が出る。一覧は増えない | 高 |
| C-02 | 工数の入力チェック | 同上 | 工数に 0.5 を入れて追加を押す | 一覧が1件増え、入力欄が空に戻る | 高 |
| C-03 | 工数の入力チェック | 同上 | 工数を空のまま追加を押す | 「入力してください」が出て、カーソルが工数欄に移る | 中 |
| C-04 | 作業種別の絞り込み | 一覧に初期データが表示されている | 作業種別を「実装」に切り替える | 実装の行だけが残る。すべてに戻すと元の件数に戻る | 高 |
```

解説

改善サイクル

1回目の出力は下書きです。抜けを足して2周まわします



> 4から1に戻るときは「作り直して」と言いません。「既存の表に追記して」と書きます。

ワーク

演習 観点表とケース表

午前に足した機能を1つ選び、観点表とケース表を自分で作ります

どんな場面か

派遣先で最初に任されるテスト設計は、たいてい他人が作った画面の一部です。仕様書は途中までしかなく、動く画面だけが手元にあります。その状態から観点を出して、他の人が実行できる形のケース表に落とすところまでを、配布の作業記録アプリで一度通します。

やること

1. 対象を1つに絞ります。午前に機能追加した配布アプリから、作業種別の絞り込み、作業種別ごとの工数合計、工数の入力チェックのどれか1つを選びます。画面全体は対象にしません
2. 対象部分のHTMLをチャットに貼り、「観点を出させる文」のページの文で観点だけを出させます。返ってきた観点に、自分の観点を2つ足します
3. 観点ごとに同値クラスと境界値を出させます。項目、同値クラス、境界値の3列で頼みます

チャットに入れてよいのは配布アプリの中身だけです。客先名、製品名、実データは入れません。

ワーク

演習 観点表とケース表（続き）

やること

1. 「ケース表の列」のページの見出し行をそのままコピーして渡し、Markdownの表で出させます。デスクトップの tpro-work フォルダの中に testcases.md という名前で保存します
2. 改善サイクルを2周します。表にない操作を自分で探して1文で書き、既存の表に追記させます
3. どの一文が効いたと思うかを1行だけ、デスクトップの tpro-work フォルダの中の prompt-memo.txt に書きます

できあがるもの

デスクトップの tpro-work フォルダの中に testcases.md（ケース表）と prompt-memo.txt（1行メモ）。今日は実行しません。実行はDay3です。

ケース表の達成基準

行数ではなく、自分が足した行と判定できる期待結果で見ます

達成の目安

- testcases.md に10行以上のケースがあり、そのうち2行以上は自分が抜けに気づいて足した行である
- どの行にも前提、操作、期待結果がそろっており、期待結果は画面に何が出るかで書かれている（正常に動作する、は不可）
- 優先度が全部「高」になっていない。落ちたら業務が止まるものだけが高になっている

つまずいたら

- 対象が絞れず観点が30個出た。入力欄1つ、選択欄1つまで絞り直します
- 午前の機能追加が途中。絞り込み欄が画面に出ているところまでを対象にしてかまいません。期待結果は自分が決めた仕様で書きます
- 観点だけ頼んだのにケース表が返ってくる。「観点だけを箇条書きで、表にしないで」を書き足します
- Markdownの表が崩れて見える。Ctrl と Shift と V でプレビューを開くと整形された形で見えます
- ケースが5行しか出ない。条件の表に戻り、境界値の列から拾い直します

ケース表の達成基準（続き）

時間が余ったら

- 登録してから削除、削除してから同じ内容で再登録、の順に操作するケースを3行足します
- 優先度が高の行だけを抜き出した実行順リストを、別の表としてもう1つ作ります
- 作業種別の絞り込みと工数の入力チェックを組み合わせたケースを2行足します

株式会社ギブリー (Givery, Inc.)
〒150-0036 東京都渋谷区南平台町15-13 帝都渋谷ビル8F

解説

人が探す抜け

AIが出しにくい抜けは、順序と前の状態に集まります

AIが1回で出す範囲

1操作ずつの独立したケースです。渡したHTMLは1画面の止まった状態なので、時間の経過や操作の順番は材料に含まれていません。だから出てきません。

人が切り口を渡すと出る範囲

登録してから削除する、前の入力が残ったまま次を開く、タブを2つ開いて交互に操作する。順序と状態を文章で渡せば、行を書くのはAIができます。

この110分の結論

ケース表の価値は行数ではなく、期待結果が判定できるかで決まります

> **AIが出すのは候補**
> **通す基準を書くのは自分**
>
> 110分でやったのは、候補を早く出して、他人が実行できる形に直す往復です。行数を稼ぐ作業ではありません。

データと報告書への橋渡し

次はデータと報告書。実行はDay3なので、今日は設計した中身をそのまま報告します

ここまでにやったこと

ケース表ができました。工数の境界値と、自分で見つけた抜けが行になっています。まだ1件も実行していません。

これからやること

ここからの70分で、ケース表に対応するテストデータをCSVで30行作り、ケース表の件数を報告書にしてPDFで出します。追加のインストールは要りません。

そのあとどこで効くか

報告書をブラウザの印刷でPDFにする経路は、テスト以外の報告書でもそのまま使えます。Day3ではこの表を実際に動かして、合否の欄を埋めます。

今日の報告書に合否の欄は作りません。未実施の理由に「Day3で実行予定」と書きます。

07

テストデータと報告書

件数と分布を指定してCSVを作り、報告書をブラウザの印刷でPDFにできるようになります

実演

データを出させる文

件数と分布を指定しないと、似た行ばかり返ってきます

ケース表を先に渡してから頼みます。列は配布アプリが内部で持っている項目名にそろえます。

```
次のテストケース表に対応する入力データを CSV で30行作ってください。
条件:
- 列は date,kind,memo,hours の4つ
- 正常系20行、境界値5行、異常系5行の内訳にする
- hours には 0、0.5、8、24 を必ず1行ずつ含める
- kind は 設計 / 実装 / レビュー / 試験 の4種だけにする
- memo は架空の作業内容にし、20文字以内にする
- date は YYYY-MM-DD でそろえる
- 実在しそうな企業名や人名は使わない
- 先頭行はヘッダにし、コードブロックだけを出力する
```

解説

内訳の3つの言葉

正常系と境界値と異常系は、同値クラスと境界値の言い換えです

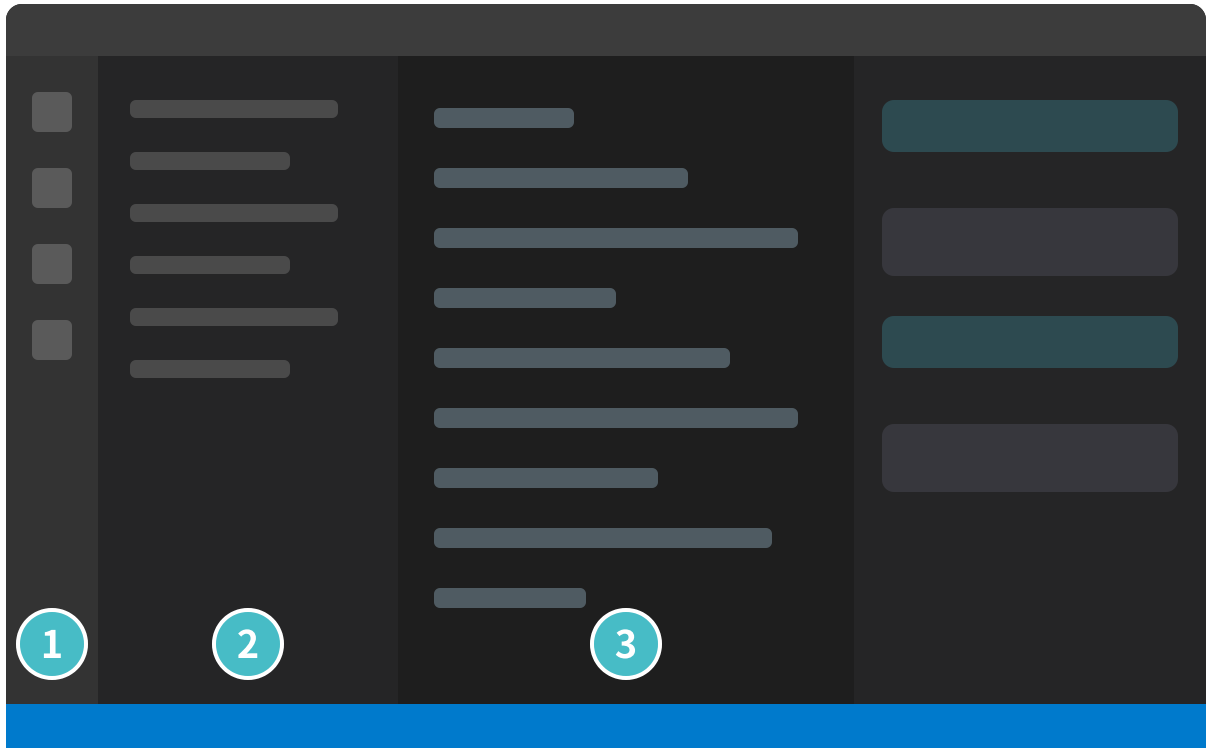
正常系	どの同値クラスにも素直に収まる値。hours なら 8 のような真ん中の値です
境界値	かたまりの境目の値。hours なら 0.5 と 24 の2つです
異常系	同値クラスの外にある値。hours なら 0 や 24.5、それに空欄と全角数字です

内訳を指定しないと、正常系ばかりの当たり障りのないデータが返ってきます。

補足

CSVの置き場所と数え方

作ったCSVは tpro-work フォルダに置き、VSCodeで行数を数えます



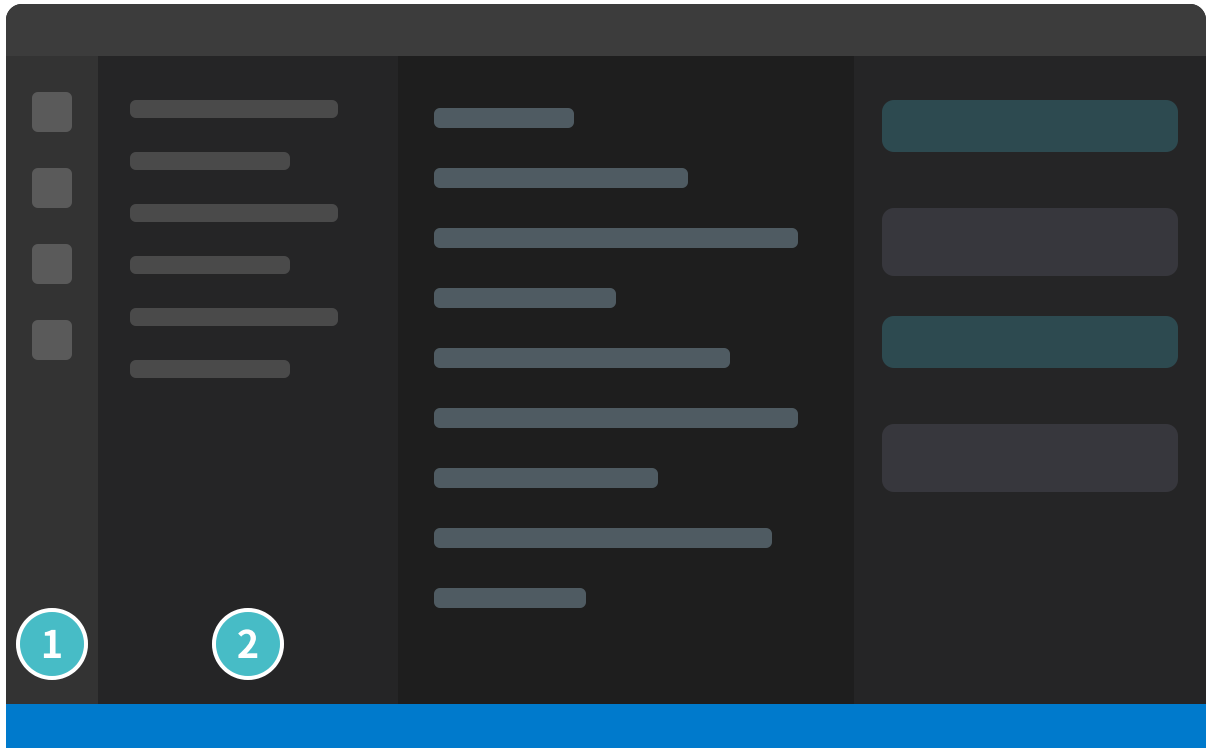
1 **アクティビティバー**
エクスプローラーとチャットの切り替えはここです。
作業中は行き来します

2 **エクスプローラー**
tpro-work フォルダの中に data.csv が並びます。
ここに出てこなければ、保存先が別のフォルダになっています

3 **エディタ**
data.csv を開くと左端に行番号が出ます。31行（ヘッダ1行＋データ30行）かを目で数えます

補足

CSVの置き場所と数え方（続き）



1 **チャットパネル**
行数が足りないときは、この欄に「異常系があと3行足りません。追加してください」と続けて書きます

2 **ステータスバー**
右側にカーソルの行番号が出ます。最終行にカーソルを置けば、行数がそのまま読めます

ワーク

演習 テストデータ生成

ケース表に対応するCSVを30行作り、内訳を自分で数えます

どんな場面か

テストデータを人が手で作ると、正常系ばかりの30行になります。境界値の行は面倒なので後回しになり、結局そこが漏れます。件数と分布を先に指定して作らせる書き方を、配布アプリの列で一度やります。

やること

1. 「データを出させる文」のページの文をチャットにコピーします。前のスレッドにケース表が残っていれば、同じスレッドで続けます
2. hours の条件を自分のケース表に合わせて直します。0、0.5、8、24 のうち、自分の表に出てこない値があれば足します
3. 返ってきたCSVをコピーし、VSCodeで新規ファイルを作って、デスクトップの tpro-work フォルダの中に data.csv として保存します

ワーク

演習 テストデータ生成（続き）

やること

1. 行数を数えます。31行になっていなければ、足りない内訳を名指しして追加を頼みます
2. hours 列に 0 と 24 が入っているか、kind 列が 設計 / 実装 / レビュー / 試験 の4種だけかを目で確認します
3. 指定と違った項目を1行だけ、tpro-work フォルダの中の data-memo.txt に書きます。何行足りなかったか、どの条件が無視されたかを書きます

できあがるもの

デスクトップの tpro-work フォルダの中に data.csv（31行）と data-memo.txt（1行）。

データ生成の達成基準

31行そろい、hours に 0 と 24 が入っていれば達成です

達成の目安

- data.csv が31行（ヘッダ1行＋データ30行）ある
- hours 列に 0、0.5、8、24 が各1行以上入っている
- memo に実在しそうな企業名や人名が1件も入っていない
- kind が 設計 / 実装 / レビュー / 試験 の4種に収まっている

つまずいたら

- 30行と頼んで15行で止まる。「残りも続けて」で続きが出ます。出た行数は毎回数えます
- コードブロックの前後に説明文が付く。手で消してかまいません
- hours に 1.25 のような 0.5 刻みでない値が混ざる。「0.5刻みだけにしてください」を足します
- 保存したのにエクスプローラーに出てこない。保存ダイアログの場所が別フォルダです。デスクトップの tpro-work を選び直します

時間が余ったら

- 同じ作業種別で同じ日に2件登録した行を3行足すよう頼み、重複を含むデータに広げます
- 作業種別ごとの hours の合計をAIに計算させてから、1種類だけ自分で検算します
- date に来月の日付を3行混ぜるよう頼み、未来日のデータを足します

解説

報告書に書く5つ

今日の報告書は、実行結果ではなく設計したケースの件数を書きます

1. **1 概要**
   何のテストの設計なのかを2行。対象のアプリと、いつ設計したかを書きます
2. **2 対象機能**
   1つだけ書きます。作業種別の絞り込み、作業種別ごとの工数合計、工数の入力チェックのどれかです
3. **3 設計したケース件数**
   合計と、優先度別の内訳。testcases.md を数えた数を書きます
4. **4 未実施と理由**
   今日は全件未実施です。理由に「Day3で実行予定」と書きます
5. **5 所見**
   1行でよいです。例として、順序に関する観点が2件抜けていたため人が追記した

> 合否の欄は作りません。実行していないので、書ける数字がありません。

実演

報告書を書かせる文

見出しの順を指定し、合否の欄を作らないと書きます

testcases.md をチャットに添付してから頼みます。件数は必ず後で自分が数え直します。

```
添付の testcases.md をもとに、テスト設計の報告書を Markdown で書いてください。
見出しは 概要 / 対象機能 / 設計したケース件数 / 未実施と理由 / 所見 の順にしてください。
条件:
- 設計したケース件数には、合計と優先度別（高・中・低）の内訳を表で入れる
- 未実施と理由には「全件未実施。Day3で実行予定」と書く
- 所見は3行以内にする
- 合否や不具合件数の欄は作らない
```

> 出てきた件数はそのまま信じません。testcases.md の行を数えて突き合わせます。

実演

PDFまでの4手

report.md からPDFまで、追加のインストールなしで4手です

1 report.md

デスクトップの tpro-work フォルダの中に保存します

→

2 report.html

「この Markdown を1枚の HTML にして、印刷用の CSS も入れてください」と頼みます

→

3 ブラウザで開く

report.html をブラウザのウィンドウにドラッグして落とします

→

4 Ctrl と P

送信先を「PDFに保存」にして、tpro-work フォルダの中に report.pdf として保存します

印刷用のCSSには、表の行が途中で分かれない指定と、余白の指定を入れてもらいます。

ワーク

演習 報告書のPDF出力

ケース表を報告書にして、PDFで出すまでを通します

どんな場面か

設計だけで終わった作業は、上に報告するときにいちばん説明しにくいところです。件数と未実施の理由を書いた1枚があれば、進捗の説明が口頭で済みます。ケース表を報告書にしてPDFにするところまでを一度通します。

やること

1. 「報告書を書かせる文」のページの文で report.md を書かせ、デスクトップの tpro-work フォルダの中に保存します
2. testcases.md を開いて行を数え、報告書の合計件数と優先度別の内訳が一致しているか突き合わせます。違っていたら手で直します
3. 「この Markdown を1枚の HTML にして、印刷用の CSS も入れてください」と頼み、tpro-work フォルダの中に report.html として保存します

報告書に客先を特定できる語が入っていないかを、保存する前に自分で見直します。

ワーク

演習 報告書のPDF出力（続き）

やること

1. report.html をブラウザのウィンドウにドラッグして開き、Ctrl と P を押します。送信先を「PDFに保存」にして、tpro-work フォルダの中に report.pdf として保存します
2. プレビューで表の右端と見出しの位置を見ます。切れていたら用紙を横向きにするか、文字を小さくするCSSを足すよう頼みます

できあがるもの

デスクトップの tpro-work フォルダの中に report.md、report.html、report.pdf の3つ。

報告書の達成基準

件数が自分の数えた数と一致し、表が切れていなければ達成です

達成の目安

- A4の1枚か2枚のPDFになっており、表の右端が切れていない
- 見出しがページの最終行に取り残されていない
- 報告書の合計件数と優先度別の内訳が、testcases.md を数えた数と一致している
- 未実施と理由に「全件未実施。Day3で実行予定」と書かれている

つまずいたら

- 印刷プレビューが真っ白になる。詳細設定の「背景のグラフィック」にチェックを入れます
- report.html をダブルクリックすると別のブラウザで開く。ウィンドウにドラッグして開きます
- AIが書いた件数が実際と合わない。よくあります。数えた数で上書きし、合わなかったことを所見に書いてもかまいません
- 表が横に切れる。まず用紙を横向きにし、それでも切れるなら列を減らします

時間が余ったら

- 優先度が高の行だけを抜き出した表を、報告書にもう1枚足します
- 報告書の冒頭に3行の要約（対象機能、設計件数、人が追記した件数）を付けます
- 同じ report.md から、社内向けと客先向けで文の硬さを変えた2種類を作らせて読み比べます

補足

PDF化の2つの道

今日はブラウザの印刷で完結します。拡張機能は持ち帰り情報です

やり方	今日の扱い	詰まりやすい点
ブラウザの印刷	今日やる方法。準備は要りません	改ページの位置はCSSで調整します。表の右端が切れやすいです
Markdown PDF 拡張	持ち帰り情報。インストールの可否は職場のルールに従います	設定を入れないと変換用のブラウザを取りに行き、社内ネットワークでは止まったままになります

> 拡張を使う場合は、設定の markdown-pdf.executablePath に、PCに入っている Edge の実行ファイルのパスを入れます。

次回持参と手元に残るもの

次回持参は tpro-work フォルダー一式です。それ以外は持ってこなくてよいです

手元に残るもの（持参不要）

model-memo.txt、prompt-memo.txt、data-memo.txt、report.md、report.html、report.pdf。自分で読み返す用です。持ってこなくてもDay3は進みます。

次回持参（必須）

tpro-work フォルダー一式。中に index.html、.github/copilot-instructions.md、testcases.md、data.csv が入っていることを画面で確認してください。

ケース表が残っていれば、Day3は続きから始められます

> **今日はAIに候補を出させた**
> **通す基準は自分で書いた**
>
> Day3はこのケース表を単体テストとE2Eに落として、ブラウザの中で実際に動かします。セキュリティの観点も足します。