Givery

HANDS-ON GUIDE ／ ハンズオンガイド

情報系 Day3 ハンズオンガイド

AI協働開発ハンズオン講座　情報系トラック　Day3

> Day3で手を動かす演習の手順を、この1冊にまとめています。投影スライドは流れていくので、手が止まったときはこちらに戻ってきてください。演習ごとに、なぜやるか、正確な操作、自分で確認する方法、詰まったときの戻し方を書いています。

> 今日の成果物は、すべてデスクトップの tpro-work フォルダの中に置きます。ダウンロードフォルダのまま作業すると、午後のフィードバックでファイルを見つけられなくなります。

今日の位置づけと1日の流れ

Day1では仕様5行から動くWebアプリを作りました。Day2では既存の作業記録アプリに機能を足し、テスト観点を表に起こしたところで止まっています。Day3は、その観点表を受入基準まで書き直し、ブラウザだけで走る単体テストに変えます。午後は認定の時間です。

午前 [180min]

時間	内容	手を動かすか
[10min]	Day2振り返りと今日の概要	聞く時間です
[60min]	テスト設計の基本とAIの役割	聞く時間です
[10min]	休憩	戻る時刻はホワイトボードに出ます
[100min]	仕様書からケース設計、単体テスト、セキュリティ観点	演習1から演習3

午後 [300min]

時間	内容	手を動かすか
[70min]	E2Eテストのデモと最新テクニック	最後に演習4だけ
[10min]	休憩	戻る時刻はホワイトボードに出ます
[40min]	認定ガイダンスとリテラシーチェック	選択式20問に回答します
[110min]	実技課題	ガイダンス[20min]と作業[90min]
[60min]	フィードバックと修了証	1人ずつ画面を見せます
[10min]	クロージング	聞く時間です

作業[90min]の間は、各自のタイミングで席を外して構いません。全体で止める休憩はそこから先ありません。

使うもの

展開先

配布ZIPの中身は、デスクトップに tpro-work という名前のフォルダを作って、その中へすべて展開します。Day1・Day2で tpro-work を作っている方は、そのフォルダに上書きで展開してください。展開したらVSCodeの「フォルダを

開く」から tpro-work フォルダを開きます。ファイルを1つだけ開くと、同じフォルダにあるライブラリの読み込みが通りません。

配布ファイル

ファイル名	役割
index.html	作業記録アプリ。午前の題材です。日付・作業種別・内容・工数(時間)の4欄を持ちます
validate.js	工数(時間)欄の判定 validateHours(value)。index.html と自分で作る tests.html の両方から読み込みます
review_target.js	演習3で読むファイル。外部入力を扱う箇所が入っています。どのHTMLからも読み込んでいません
template.html	実技課題のひな形。複製して kadai.html という名前に変えて使います
sample_work_log.csv	実技課題で読み込む作業記録。30行5列、文字コードはUTF-8です
papaparse.min.js	CSVを読むライブラリ。実技課題で使います
chart.umd.js	棒グラフのライブラリ。任意要件のときだけ使います
02_リテラシーチェック.html	午後のリテラシーチェックで開きます
README.txt	展開先と開き方の説明です

papaparse.min.js と chart.umd.js は同梱しています。ネットから取得する必要はありません。HTMLからは同じフォルダのファイル名で読み込むので、フォルダを分けると読み込みが通りません。

今日この中に自分で作るファイル

ファイル名	どこで作るか
cases.md	演習1
tests.html / tests.js	演習2
security_review.md	演習3
fail_note.md	演習4
kadai.html	実技課題。template.html の複製です
furikaeri_（自分の名字ローマ字）.md	実技課題の提出物

いずれも配布物には入っていません。すべて tpro-work フォルダの中に作ります。

HTMLをブラウザで開く手順

1. エクスプローラーでデスクトップの tpro-work フォルダを開きます
2. 開きたいHTMLファイルをダブルクリックします
3. 既定のブラウザで画面が開きます。ブラウザの種類は問いません
4. VSCodeでファイルを直したら、Ctrl+S で保存してからブラウザを再読み込み（F5）します

ダブルクリックで別のアプリが開く場合は、ファイルを右クリックして「プログラムから開く」からブラウザを選びます。保存せずに再読み込みしても画面は変わりません。

午前の講義で押さえること

演習1から演習3は、ここに書いた考え方をそのまま使います。手が止まったらこの節に戻ってください。

通過と充足は別のこと

テストが通ったというのは、実装の振る舞いと自分が書いた期待結果が一致した状態です。期待結果の出どころは問われていません。実装を読んで期待結果を書けば、必ず全件通ります。件数を増やしても、通る件数が増えるだけです。

仕様を満たしたというのは、仕様に書かれた合格条件と実装の振る舞いが一致した状態です。判定の元は仕様書と受入基準で、実装は根拠に使いません。1件でも合わなければ、実装かテストのどちらかが仕様から外れています。

今日の題材でいうと、工数の上限は0.5以上24以下という仕様ですが、配布した validate.js は 240 で判定しています。実装を読んで期待結果を書くと、この食い違いは一度も表に出ません。

受入基準は数えられる形まで書き直す

手順	やること
対象を1つに絞る	画面ではなく、入力欄1つか関数1つまで絞ります。今日は工数(時間)の欄だけです
合格を数字で書く	0.5以上24以下、小数点は1つまで。適切な値、妥当な範囲、正しい入力は基準になりません
不合格を列挙する	0、24.5、全角の８、空文字、abc、1..5。落ちてほしい入力を先に書き出します
どうなるかを書く	false を返す、一覧に行が追加されない。判定できる形で1つ決めます

書き直しの一例です。「工数には適切な値を入力できること。不正な値はエラーとすること」という2行からは、ケースを1件も起こせません。「工数(時間)は半角数字で0.5以上24以下、小数点は1つまで。範囲外・全角数字・空文字・数字以外は false を返し、一覧に行を追加しない」まで書けば、境界が4つ、異常が4つ、そのまま数えられます。

観点の型5つ

観点	狙い	工数(時間)欄での例
同値	同じ扱いになる範囲から代表を1つ選ぶ	8 を入れる
境界	範囲の端と、その1つ外側を狙う	0.5 / 24 / 0.4 / 24.5

観点	狙い	工数(時間)欄での例
異常	型や書式が違う入力を渡す	全角の８ / 空文字 / abc / 1..5
状態	操作の順番や回数で変わる挙動を見る	空のまま追加ボタンを2回続けて押す
表示	画面に出た結果の見え方を見る	0.5 が 0.5 と出るか、追加した行が末尾に付くか

型ごとに最低1件を作れば、抜けの大半は防げます。それ以上は工数と相談です。

人とAIの分担

基準を決めるのは人、候補を並べるのはAI、残す行を選ぶのは人、コードに変えるのはAIです。この順番は入れ替えません。合格と不合格の境目は、仕様書を読んだ人にしか決められません。

理解度チェック1

AIが出したテストケース表が返ってきました。最初にやることはどれですか。

- A. 件数が少ないので、追加の候補を出させる
- B. 期待結果の列を、受入基準と1件ずつ突き合わせる
- C. テストコードに変えて、まず全件走らせてみる
- D. カバレッジを測って、80%に届いているか確認する

> 自分で選んでから読んでください。正解はBです。期待結果が実装から写されていると、全件PASSでも仕様との差は一度も表に出ません。表の状態で突き合わせるのが一番安く済みます。Aの件数は品質の指標になりません。Cは順番が逆で、期待結果が間違っていれば結果も間違います。Dは通った行の割合しか測れず、今日の validateHours は3行なので1件走らせただけで100%になります。突き合わせは全件やらなくて構いません。境界と異常の型に付いた4件だけで、写しかどうかは分かります。

演習1 ケース表の作成 [約20min]

どんな場面か

客先から渡された仕様書に「工数には適切な値を入力できること」と1行だけ書いてあります。この1行では期待結果が書けないので、まず数えられる形に直してからケース候補を出します。派遣先で最初に頼まれるのは、たいていこの書き直しの部分です。

題材と仕様

項目	内容
対象	作業記録アプリ index.html の工数(時間)欄の入力チェック。判定は validate.js の validateHours(value) が持ちます
合格条件	半角数字で0.5以上24以下。小数点は1つまで。合格なら true を返します

項目	内容
不合格の扱い	false を返し、一覧に行を追加しません。エラー文の文言は今日は問いません
やらないこと	画面の見た目、日付欄と内容欄と作業種別欄のチェック、サーバー側の処理

やること

1. tpro-work フォルダに cases.md を新規作成し、先頭に受入基準を3行で書きます。半角数字だけ、0.5以上24以下、小数点は1つまで、の3点を数字を入れて書きます。エクスプローラーの tpro-work の行を右クリックして「新しいファイル」からでも作れます
2. Copilot Chat を Ctrl+Alt+I で開き、下の依頼文をそのまま貼ります。【受入基準】の下は、自分が書いた3行に差し替えます
3. 返ってきた表を cases.md に貼り付け、業務上あり得ない行と、同じ扱いになる重複行を消して8件から12件に絞ります
4. 境界と異常の型に付いた4件だけ、期待結果を受入基準と突き合わせて、違っていれば自分で直します
5. 消した行を1つ選び、なぜ消したかを cases.md の末尾に1行書きます

```
次の受入基準に対するテストケース候補を表で出してください。
列は 観点 / 入力 / 期待結果 の3列です。
観点は 同値・境界・異常・状態・表示 の5種類から選んでください。

【受入基準】
工数(時間)は半角数字で0.5以上24以下、小数点は1つまで。
範囲外・全角数字・空文字・数字以外は false を返す。
不合格のとき一覧に行は追加されない。

実装は見ずに、この基準だけから作ってください。
```

最後の1文が無いと、実装をなぞった期待結果が返ります。ここは省かないでください。

できたかの確認

cases.md を開いて、次の4点を自分で見ます。件数ではなく中身で見ます。

- 観点の列に、5種類のうち3種類以上が入っている
- 境界の型に 0.4 / 0.5 / 24 / 24.5 の4件が入っている
- 期待結果の列に「適切」「妥当」「正しく」という語が1つも無い
- 末尾に、消した行の理由が1行ある

うまくいかないとき

症状	戻し方
8件に届かない	異常の型が抜けています。全角の８・空文字・abc・1..5 の4件を出させてください。状態の型は今日の題材では作りにくいので勧めません
20件以上返って絞れない	「同じ扱いになる入力は1件に寄せてください」と追加で頼みます
期待結果が全部「エラー」で埋まっている	false を返すのか、行が追加されないのか、どちらを見るのかを1件ずつ決めます
実装のコードを読んだ期待結果が返ってくる	依頼文の最後の1文が抜けています。貼り直します

時間が余ったら

- 同じ受入基準を英語で貼り直して、返ってくる候補の粒度が変わるか比べます。cases.md には反映しません
- 日付欄の受入基準を3行だけ書いてみます。ケース表は作りません

演習2 ブラウザでテストを走らせる [約22min]

どんな場面か

客先から「テストをやった証拠を見せてほしい」と言われる場面を想定します。テストランナーを入れられない現場でも、ブラウザで開いてPASSとFAILが並んだ画面をそのまま見てもらえます。今日はその画面を自分の手で作ります。

書くのはVSCode、動かすのはブラウザです。この2つを行き来します。

使う部品

tests.html は次の6行です。tpro-work フォルダに、index.html と validate.js と同じ階層で作ります。ul の id は result で固定します。

```
<!doctype html>
<meta charset="utf-8">
<h1>テスト結果</h1>
<ul id="result"></ul>
<script src="validate.js"></script>
<script src="tests.js"></script>
```

合否を判定して画面に1行ずつ足す関数は、次の形です。写経は要りません。演習2ではCopilotに書かせます。この10行が、Jestなどのテストランナーの一番外側にあたる部分です。

```
const RESULT = document.getElementById('result');
function check(name, actual, expected) {
  const ok = actual === expected;
  const li = document.createElement('li');
  li.textContent = (ok ? 'PASS' : 'FAIL') + ' ' + name;
```

```
  li.style.fontWeight = ok ? 'normal' : 'bold';
  RESULT.appendChild(li);
}
check('8は合格', validateHours('8'), true);
```

やること

1. tpro-work フォルダに tests.html を、上の6行で作ります。id は result のままにします
2. cases.md を開いた状態で Copilot Chat に、次の依頼文を貼ります
3. 返ってきたコードを tests.js として tpro-work フォルダに保存します
4. tests.html をエクスプローラーから右クリックしてブラウザで開きます
5. FAILが出た行を1件選び、直すのは実装かテストかを cases.md の末尾に理由付きで1行書きます

```
この表の各行を check(name, actual, expected) の呼び出しに変えて
tests.js を作ってください。check の定義は書かないでください。
```

「check の定義は書かないでください」を落とすと、tests.js の中に check がもう1つできて二重定義になります。定義は tests.html 側か別ファイルに1つだけです。

できたかの確認

全件PASSは未達です。太字のFAILが1件出ていることを確認します。

- 太字のFAILが1件出ている。24.5 を不合格としたケースが落ちます
- PASSの件数が、cases.md のケース件数から1を引いた数と一致している
- cases.md の末尾に、実装かテストかどちらを直すかの判断が1行ある

> FAILの原因は、自分で1件見てから読んでください。配布した validate.js の判定は 0.5以上240以下 になっています。仕様の上限は24なので、実装のほうが仕様から外れています。受入基準どおりに期待結果を書いた場合、24.5 の行で必ずFAILを引きます。

全件PASSは、テストが何も見ていない可能性を含みます。落ちるはずの入力で落ちるかどうかを、必ず1件は確かめます。

うまくいかないとき

症状	戻し方
画面が真っ白	tests.html と validate.js が同じ階層にありません。script の src とフォルダの中身を見比べます
全件PASSで終わった	期待結果が実装から写されています。24.5 の行の期待結果を false に直して走らせ直します
一覧が出ない	F12 のコンソールに getElementById が null と出ていれば、ul の id が result になっていません

症状	戻し方
validateHours が未定義と出る	script の順番が逆です。validate.js を tests.js より先に置きます
手が完全に止まった	check の呼び出しを1行だけ手で書いて動かします。1行動けば残りは流れます

時間が余ったら

- 件数を画面の上に「12件中11件PASS」の形で出す行を足します。tests.html だけで完結し、後続には影響しません
- validate.js の上限を仕様どおり24に直し、FAILが消えることを確かめます。確かめたら240に戻しておきます

演習3 セキュリティ観点のレビュー依頼 [約8min]

どんな場面か

客先から他人が書いたファイルを渡されて、目を通しておいてほしいと言われる場面を想定します。全部読む時間が無いときほど、観点を2つに絞って機械的に確認する頼み方が効きます。今日はその頼み方を1回やってみます。

見る2つの観点

入力を画面に出す書き方が1つ目です。`el.innerHTML = '結果: ' + input;` と書くと、入力に `<img src=x onerror=alert(1)>` が入っていたときにタグとして解釈されて実行されます。`el.textContent = '結果: ' + input;` ならタグは文字として表示されるだけです。配布アプリの描画処理は td.textContent を使っているので、すでにこの形になっています。

SQL文の組み立てが2つ目です。今日の題材にDBはないので、概念だけ扱います。危ういのは、文字列を足して命令を作る書き方そのものです。

```
// 危うい書き方（入力が命令の一部になる）
const sql = "SELECT * FROM logs WHERE user = '" + name + "'";
// name に  ' OR '1'='1  が入ると WHERE の条件が消えます

// 値を分けて渡す書き方
const sql = "SELECT * FROM logs WHERE user = ?";
db.query(sql, [name]);
```

攻撃文字列を自分で作って試すことはしません。仕組みを知りたい場合は OWASP Top 10 のページを見てください。

やること

1. tpro-work フォルダの review_target.js を開きます。このファイルには外部入力を扱う箇所が2件あります。対象を index.html にすると、td.textContent で描画しているため「該当なし」が返って空振りします
2. Copilot Chat に次の依頼文を貼ります
3. 返ってきた指摘の該当行を必ず開いて目で見ます。実際に外部入力が入る経路になっているものと、そうでないものを分けます

4. 指摘の一覧と自分の判断を security_review.md に書いて tpro-work フォルダに保存します

```
このファイルを2つの観点だけで見てください。
観点1、外部入力の文字列を innerHTML に渡している箇所。
観点2、文字列連結でSQL文を組み立てている箇所。
該当箇所を行番号付きで挙げ、直し方を1行で示してください。
```

手順3が本題です。指摘された箇所のうち、実際に利用者の入力が届くのはどこか。届かない箇所は、いま直す必要がありません。この判断はAIが代わりにやってくれません。

できたかの確認

- innerHTML に渡している箇所と、SQL文を文字列連結している箇所の2件が、行番号付きで挙がっている
- security_review.md に、2件それぞれ外部入力の経路かどうかの判断が1行ずつある

うまくいかないとき

症状	戻し方
「該当なし」が返る	対象ファイルが review_target.js になっているか確認し、観点1と観点2を別々に投げ直します
指摘が10件返って読めない	行番号順に上から3件だけ見る、と絞ります
提案をそのまま適用してファイルが壊れた	Ctrl+Z で戻します。適用する前に差分を目で見ます

時間が余ったら

- AIが出した書き換えを1件だけ実際に適用し、ブラウザで開いて表示が壊れていないか確かめます
- 同じ依頼文から「行番号付きで」を外して投げ、指摘の使いにくさがどう変わるか比べます。ファイルには反映しません

午前の到達点

tpro-work フォルダに cases.md、tests.html、tests.js、security_review.md の4つが残っていれば午前は達成です。午後の実技課題で開き直すので消さないでください。昼休憩の間も、VSCodeとブラウザ、cases.md と tests.html は開いたままにします。

持ち帰る考え方は1つです。期待結果の出どころは仕様であって実装ではない。全件PASSを見たら、落ちるはずの入力で落ちるかを1件確かめる。

午後のデモで見ること

E2Eテストのデモは、Node.js のインストールが要るツールを使うため、貸与PCでは動かせません。画面を見る時間です。写す必要はありません。

単体テストとE2Eの違い

上下ではなく、落ちたときに疑う範囲の広さが違います。作業記録アプリでいうと、追加ボタンを押す、入力を受け取る処理、入力チェックの関数、表に1行増える、の4段階があります。午前の単体テストが囲むのは3番目だけです。E2Eは1番から4番までをまとめて囲みます。

層	確かめること	件数の考え方
単体テスト	関数に値を入れて戻り値を見る	一番多く置きます。午前に書いたのがここです
結合テスト	画面と処理のつなぎ目	今日の環境ではブラウザで開いて手で確認します
E2Eテスト	操作の流れ全体	止まると業務が回らない道筋の数だけに絞ります
探索的テスト	人が触って気づくこと	自動化しません。件数では数えません

E2Eは増やすほど壊れます。画面の文言を1つ変えただけで落ちるためです。修正が積もって誰も直さなくなった状態が、現場でよく見る落ちっぱなしのテストです。

デモで使う Playwright は、ブラウザでの操作を記録してテストコードに書き出すツールです。落ちた時点のスクリーンショットと操作の記録がファイルに残ります。持ち帰り先は https://playwright.dev/docs/codegen です。

失敗出力の貼り方

落ちた出力を全文貼ると、端末名、利用者名、社内のフォルダ構成、案件名が一緒に流れます。派遣先の環境では、ここが機密の線に触れます。貼るのは次の4か所だけです。パスは末尾のファイル名だけに書き換えます。この4か所に絞っても、返ってくる見当の精度は落ちません。

- テスト名
- 期待と実際
- 止まった場所
- 添付の有無

技術の仕分け

技術	できること	今日の扱い
エージェントモード	複数ファイルの編集と確認を繰り返す	実技課題で使えます
カスタム指示ファイル	前提を常に効かせて指示を短くする	Day2で作った続きが使えます
コードレビュー支援	変更した差分に指摘をもらう	実技課題で使えます

技術	できること	今日の扱い
E2E自動テスト	画面操作を記録して繰り返し実行する	デモのみ。Node.js が要ります
Model Context Protocol	AIから社内の検索やチケット管理を呼び出す	紹介のみ。接続先ごとに設定ファイルが要ります

最初に手を付けるならカスタム指示ファイルです。設定は一度で済み、効果が毎回出ます。

理解度チェック2

ブラウザ内のJavaScriptで単体テストを書く場合、Jestやpytestのようなテストランナーを使う場合と比べて、追加で自分が用意することになるものはどれですか。

- A. テストに使う入力値
- B. 期待する結果の定義
- C. 合否を判定してまとめて表示する仕組み
- D. テスト対象となる関数そのもの

> 自分で選んでから読んでください。正解はCです。テストランナーが受け持つのは、複数のテストをまとめて実行し、結果を並べ、落ちた場所を指すところです。午前はそこを10行ほどの検証関数として自分で書きました。Aの入力値はどちらの環境でも自分で決めます。Bの期待値は仕様から人が決めます。Dはどちらの環境でも自分で書くので差になりません。環境が変わっても、入力値と期待値を決める仕事は残ります。楽になるのは実行と集計の部分だけです。

演習4 失敗の見当出し [約3min]

どんな場面か

午前に出た FAIL が1件、まだ直っていません。現場では、落ちた原因を自分の言葉にできないまま相談すると、往復が2回3回に増えます。先に見当を3つ持ってから相談する形を、ここで1回やります。E2Eツールが無くても、この手順は今日から使えます。

3分です。文章を練らないでください。1行ずつで構いません。

やること

1. tpro-work フォルダにある tests.html をブラウザで開き、FAIL と書かれた行をコピーします。見つからないときは Ctrl+F で FAIL を検索します
2. VSCodeに戻り、Ctrl+Alt+I でチャットパネルを開きます
3. 次の3行をそのまま打ち、続けてコピーした FAIL の行を貼って送ります
4. 返ってきた3つの見当から1つ選び、tpro-work フォルダに fail_note.md を作って、選んだ見当と、直すのは実装かテストかを1行ずつ書きます

```
次はブラウザ内で動かした単体テストの失敗出力です。
原因の見当を3つ、可能性の高い順に挙げてください。
直すのは実装かテストかも書いてください。コードは書き換えないでください。
```

貼る内容は、次の形まで削ります。

```
[FAIL] 工数25は不合格になる
expected: 不合格    received: 合格
止まった場所: tests.html の check 呼び出し 7行目
添付: なし。ブラウザ内実行のためスクリーンショットは残りません
```

できたかの確認

fail_note.md は2行だけの短いファイルです。

- 原因の見当が1行、直す対象が実装かテストかが1行、合わせて2行ある
- 直す対象を選んだ理由が、午前に書いた受入基準の文と結び付いている。上限の値が仕様と実装で違う、という形で言えていれば到達です
- Copilot の返答をそのまま貼った状態になっていない。自分が選んだ1つだけが残っている

うまくいかないとき

症状	戻し方
FAIL の行が見つからない	tests.html を開いたブラウザで Ctrl+F を押し、FAIL で検索します
見当が1つしか返らない	「見当を3つ挙げてください」ともう一度だけ送ります。依頼文を作り直す必要はありません
見当が全部テストの書き方の話になる	「実装を疑う可能性を1つ入れてください」と足して投げ直します
チャットの返答が返ってこない	同じ文をもう一度送ります。ネットワークの一時的な失敗が大半です
新しいファイルが作れない	エクスプローラーの tpro-work の行を右クリックして「新しいファイル」を選びます

時間が余ったら

- 同じ FAIL を、期待と実際の行だけ抜いて貼り直し、返ってくる見当がどう変わるか比べます。fail_note.md には書き足しません
- fail_note.md の末尾に、この失敗を客先に報告するときの1文を書いてみます。ファイル名とパスを出さずに書けるかを試します

リテラシーチェック [40min]

どんな場面か

現場では調べながら仕事をします。覚えているかを測っても意味がないので、資料参照可の設計にしています。3日間でどこに何が書いてあったかをたどれるかを見ます。落とすための時間ではありません。点数は出しません。

やること

1. 配布された 02_リテラシーチェック.html を開きます。選択式20問です。回答時間は[20min]です
2. Day1からDay3のスライド、このハンズオンガイド、自分のメモ、tpro-work フォルダの中身は全部見て構いません。隣の人との相談と Copilot に聞くことだけ、この[20min]は控えてください
3. 分からない問題は空欄のまま次へ進み、戻る印を付けておきます。最後に3問だけ全体で確認します
4. 早く終わったら、実技課題の要件を先に読んでおきます

出題の範囲は、機密の伏せ方4ルール、AIの出力を確かめる手順、テスト設計の考え方、Copilot の操作の4領域です。

できたかの確認

成果物はありません。次の3点で見ます。

- 20問すべてに選択が入っている。迷った問題も空欄にせず、いちばん近いものを1つ選んで印を付けてある
- 答えの根拠にした資料の場所を、少なくとも3問について言える。ページ番号でもファイル名でも構いません
- 機密の伏せ方の設問で、丸める対象が3つではなく4つあったことに自分で気づけている

回答方法と回収方法は当日の案内に従ってください（要確認）。

うまくいかないとき

症状	戻し方
どこに書いてあるか探せない	探すこと自体は目的ではありません。分かる範囲で答えて構いません
残り[5min]で空欄が5問以上ある	選択肢を2つに絞って、どちらかを選びます。空欄のままより、選んで外したほうが次の確認で拾えます
時間内に終わらなかった	そのままで構いません。未回答の数で扱いは変わりません
設問の意味が読み取れない	その場で申し出てください。設問を言い換えます

よく外す3問

> 自分で回答してから読んでください。1つ目は機密の伏せ方です。丸める対象は、客先名と製品名とプロジェクト名と取引先名の4つです。3つと答える人が毎回います。加えて、実データと実ファイルとスクリーンショットは使いません。2つ目はAIの出力の確かめ方です。読んで筋が通っていることは、確かめたことになりません。ブラウザで開いて動かし、結果を目で見るまでが1周です。3つ目は期待値を決める人です。仕様から期待値を決めるのは人です。AIは候補を出すところまでで、正しいかどうかの判断は代わりにやってくれません。

時間が余ったら

- 迷った問題を1問選び、なぜ迷ったかを1行だけメモしておきます。フィードバックの時間で見せる材料になります
- 実技課題の必須要件を先に読み、最初に Copilot へ投げる1文を tpro-work のメモに下書きしておきます

実技課題 [110min]

ガイダンス[20min]、作業[90min]です。新しい技術は出しません。3日間で扱った内容の組み合わせだけで足ります。

どんな場面か

客先から作業記録のCSVを受け取り、中身の汚れを見つけて担当ごとに集計して報告する。常駐先に入って最初に回ってくるのはこの形の作業です。表記がそろっていない、単位が文字で混ざっている、必須の欄が空いている。3日間で別々にやってきた読み込み・検出・集計・テストを、ここで1枚のHTMLにつなげます。

必須要件と到達の判定

区分	やること	到達の判定
必須1	CSVをドロップして表に出す	30行すべてが表に並ぶ
必須2	汚れを検出して警告一覧に出す	仕込んだ4種のうち3種以上が出る
必須3	担当ごとの合計時間を出す	担当3名分が出て、手で足した数と一致する
必須4	検証関数で6件テストする	PASS/FAIL付きの6行が並び、正常系3件がPASS
任意	棒グラフ、絞り込み、CSV書き出し	判定には入りません
やらないこと	サーバー起動、追加インストール、実データ持ち込み	使いません

完成画面の並び

上から順に5つ並べば完成です。見た目は判定に入りません。色やレイアウトは触らなくて構いません。

1. CSVを落とす枠。画面の一番上に破線の四角を置き、ここにCSVを落とす、と文字を入れます
2. 読み込んだ行の表。30行が5列で並びます。列の順番はCSVと同じで構いません
3. 警告の一覧。何行目のどの列がどう変か、を1行1件で並べます
4. 担当ごとの合計。担当3名の名前と合計時間の小さい表です
5. テスト結果のリスト。先頭にPASSかFAILが付いた6行。ページの一番下で構いません

template.html には、この5つの見出しと差し込み先の div がすでに用意してあります。

読み込むデータ

tpro-work フォルダの sample_work_log.csv です。全30行、担当は3名、文字コードはUTF-8です。先頭5行は次のとおりです。

```
日付,担当,工程,作業内容,作業時間
2026/09/01,佐藤,設計,画面遷移の見直し,3.5
2026/9/2,鈴木,実装,一覧表示の修正,2
2026-09-03,佐藤,試験,結合テストの実施,3.5 h
2026/09/04,,調査,ログの調査,1.5
2026/09/05,田中,実装,入力チェックの追加,２
```

仕込んである汚れは4種類だけです。探すのはこの4つです。

1. 日付が3通りの書き方で混在
2. 作業時間が「3.5 h」の単位付き文字列
3. 担当が空欄の行が1件
4. 作業時間が全角数字の行が1件

作業[90min]の区切り

区切り	やること
[15min] 観点を出す	要件を読み、確かめることを箇条書きにします。ここではエディタを開きません
[45min] 実装する	依頼文を貼り、1つ足すたびにブラウザで開いて確かめます
[15min] テストを回す	検証関数で6件流して、PASS/FAILを画面に出します
[15min] 言語化する	提出するテキストを書きます。実装が途中でもこの[15min]は使います

実装が終わっていなくても、残り[30min]でテストと言語化に移ります。

やること

1. tpro-work フォルダを開き、template.html を同じフォルダに複製して kadai.html という名前に変えます
2. 最初の[15min]はエディタを開かず、確かめることを箇条書きにします。tpro-work の中に kanten.txt を作って書いても構いません
3. kadai.html を VSCode で開き、チャットパネルに下の依頼文をそのまま貼ります。前提3行のあとに依頼を1つだけ書きます
4. 返ってきたコードを kadai.html に貼り、Ctrl+S で保存してからブラウザで開き、sample_work_log.csv をドロップします
5. 表が出たら、警告の一覧、担当ごとの合計、検証関数6件をこの順に足します。1つ足すたびに保存してブラウザを再読み込みします
6. 残り[15min]で furikaeri_（自分の名字ローマ字）.md を作り、観点の箇条書き、効いた依頼文、外した依頼文とその対処の3つを書きます

```
前提1: 1枚のHTMLだけで動かします。Node.js とビルドは使いません。
前提2: 読み込むCSVは同じフォルダの sample_work_log.csv です。
  列は 日付,担当,工程,作業内容,作業時間 の5列、全30行です。
前提3: CSVはドラッグ＆ドロップで受け取ります。fetch は使いません。
```

```
  同じフォルダの papaparse.min.js を使い、ライブラリは増やさないでください。

依頼: kadai.html に、CSVをドロップしたら全30行を表に出す処理を追加してください。

出力: 変更後の kadai.html の全文を1つのコードブロックで出してください。
  変更した箇所を最後に1行で説明してください。
```

2手目以降も同じ形です。前提3行は毎回そのまま、依頼の1行だけを差し替えます。

前提を書かずに「CSVを読み込んで表示して」とだけ頼むと、Node.js で動かす前提のコードや、fetch でファイルを読むコードが返ります。どちらも今日の環境では動きません。貼って開いて真っ白、で[15min]溶けます。

必須4の検証関数

午前に書いた10行をそのまま使います。kadai.html の script の中に貼って、下の呼び出しを6行書けば終わります。

```
<ul id="result"></ul>
```

```
function check(name, actual, expected) {
  const ok = actual === expected;
  const li = document.createElement('li');
  li.textContent = (ok ? 'PASS' : 'FAIL') + ' / ' + name;
  document.getElementById('result').appendChild(li);
}
check('全角数字は警告になる', isDirty('２'), true);
check('半角数字は警告にならない', isDirty('2'), false);
```

作業中に見る5か所

場所	見ること
アクティビティバー	Copilot Chat のアイコン。閉じてしまったらここから開き直します
エクスプローラー	kadai.html と sample_work_log.csv と papaparse.min.js が同じ並びに見えていれば置き場所は正しいです
エディタ	kadai.html を開きます。Ctrl+S で保存してからブラウザを再読み込みします
チャットパネル	依頼文を貼る場所です。履歴はフィードバックの時間に見せるので消さないでください
ステータスバー	開いているファイルの文字コードが出ます。CSVの文字が化けたときはここがUTF-8かを見ます

できたかの確認

数で確かめます。動いた気がする、では判定できません。

- sample_work_log.csv の30行がすべて表に並んでいる。表の上に行数を出しておくと、数えずに確認できます
- 仕込んだ汚れ4種のうち3種以上が警告一覧に出ている。警告が30件出ている場合は判定が緩すぎます

- 担当3名の合計時間が画面に出ていて、電卓で足した数と一致する
- テスト結果が6行出て、各行の先頭に PASS か FAIL が付いている。正常系3件は PASS です
- furikaeri_（名字ローマ字）.md に、外した依頼文が1つ以上書かれている

提出物と伏せ方

出すのは furikaeri_（自分の名字ローマ字）.md というテキストファイル1つだけです。tpro-work フォルダの中に作ります。kadai.html と読み込んだCSVは提出しません。コードと実データは手元に残します。書く中身は、観点の箇条書き、効いた依頼文、外した依頼文とその対処の3つで、A4で1枚に収まる分量で足ります。

提出フォームの冒頭に、客先名・製品名・案件名・取引先名を業界と工程に丸める注意書きが常に出ています。貼る前に一度読んでください。貼った内容は手元のファイルにそのまま残ります。提出先の案内は当日お伝えします（要確認）。

うまくいかないとき

症状	戻し方
汚れの判定を作り込みすぎて時間が無くなる	4種のうち3種でよいので、日付の表記ゆれと担当の空欄の2つから始めます
観点出しを飛ばして実装から始めている	[5min]でよいので箇条書きに戻します。観点が無いと、必須4で何を6件テストするかが決まりません
実装が[45min]で終わらない	必須3を捨てて必須4に進みます。テストと言語化が残っているほうが到達に近いです
画面が真っ白、CSVを落としても反応しない、文字が化ける、テスト結果が出ない	下の「うまくいかないときに見るところ」を開きます
作り直したくなった	実装の[45min]の中なら構いません
サンプルCSVを増やしたい	自分で作った架空データなら構いません。実データは使いません

時間が余ったら

- 任意要件の棒グラフを1つ足します。同じフォルダの chart.umd.js を使い、担当ごとの合計を棒にします。template.html の該当行のコメント記号を外すと読み込めます
- 警告の一覧をCSVで書き出すボタンを足します。書き出したファイルは tpro-work に置きます
- 全角数字の作業時間を半角に直して集計に入れる処理を足します。直した行数を画面に出すと、直したことが自分で確かめられます

フィードバックと修了 [60min]

どんな場面か

現場で先輩に聞くときと同じ場面です。動いているところだけ見せても解決しません。動かないものと、そのとき自分が何をしたかを出せる人ほど、答えが速く返ってきます。この時間は、その出し方を練習する時間でもあります。動いていない状態のまま持ってきて構いません。

持ち時間は45分を受講者数で割った数です。15名で3分、10名で4分半、5名で9分。開始前にホワイトボードに出ます。座席の端から順に回ります。

見るもの

見るもの	用意しておくもの
動いている範囲	kadai.html をブラウザで開いた状態
詰まった箇所	チャットパネルの履歴。消さないでください
書いたテキスト	furikaeri_ の書きかけ。3行でも構いません

見た目、コードのきれいさ、必須が何個埋まったかは見ません。

やること

1. kadai.html をブラウザで開き、sample_work_log.csv を読み込んだ状態にしておきます
2. VSCode のチャットパネルを開き、直せなかった依頼文が残っている位置までスクロールしておきます
3. furikaeri_（名字ローマ字）.md を開いておきます。白紙の場合は3行だけ先に書いておきます
4. 順番が来たら、動いている部分を30秒で見せ、次に直せなかった箇所を1つだけ挙げます
5. 直し方を口で聞き、自分のキーボードで直してブラウザで結果を確かめます
6. 直った内容を furikaeri_ に1行追記します

呼ばれるまでは手を止めなくて構いません。課題の続きでも、任意要件に進んでも構いません。ブラウザ、チャットパネル、furikaeri_ の3つは開いたままにします。持ち帰りの宿題にはなりません。詰まった箇所はこの場で直します。

できたかの確認

- 見せる前に、ブラウザとチャット履歴と furikaeri_ の3つが開いている
- 詰まった箇所を1つに絞って、言葉で説明できている
- 直し方を自分のキーボードで入力し、ブラウザで結果を確かめている
- furikaeri_ に、この時間で直した内容が1行増えている

うまくいかないとき

症状	戻し方
何も動いていないので見せられない	そのまま見せます。真っ白の画面のほうが原因が早く分かります
質問が「全部分かりません」になる	直前に押した操作と、そのとき出たメッセージの2つだけ言います
詰まりが多すぎて1つに絞れない	表が出ていない、警告が出ていない、テストが出ていない、の順で上から1つ選びます
順番待ちで手が止まる	任意要件の棒グラフか、汚れの4種目に手を付けます

時間が余ったら

- sample_work_log.csv を tpro-work の中で複製し、複製した側の日付を1行だけ壊して読み込み、警告が出るか確かめます。配布ファイルは書き換えません
- furikaeri_ に「明日の現場で最初に試すこと」を1行足します
- 直せなかった箇所が残っている場合は、依頼文に前提の行を1つ足して投げ直します

修了証

修了証は全員にお渡しします。3日間の内容に自分の手で取り組んだことに対してお渡しするものです。必須要件が全部埋まっているかは条件に入っていません。この時間の最後にまとめてお渡しします。

うまくいかないときに見るところ

症状ごとに見る場所が決まっています。当てずっぽうで直さないでください。

症状	まず見る場所	直し方
画面が真っ白	F12 のコンソールの赤い行	script の src がファイル名と1文字違い。綴りを合わせます
CSVを落としても何も起きない	dragover のイベント登録	dragover 側にも preventDefault を入れます
日本語が化ける	ステータスバーの文字コード	配布の sample_work_log.csv に戻します。UTF-8です
テスト結果が出ない	ul の id と getElementById の文字列	id は result で固定。1文字でも違うと出ません
合計が合わない	単位付きと全角数字の行	数値に直してから足します。文字列のままだと連結されます

症状	まず見る場所	直し方
validateHours が未定義と出る	script の並び順	validate.js を tests.js より先に置きます
ライブラリが読み込めない	フォルダの階層	HTMLと同じフォルダに papaparse.min.js が並んでいるか確認します
Copilot Chat が見当たらない	アクティビティバー	Ctrl+Alt+I で開き直します
保存したのに画面が変わらない	保存とブラウザの更新	Ctrl+S で保存してから F5 でブラウザを再読み込みします
チャットの返答が返ってこない	送信し直す	同じ文をもう一度送ります。時間内は先に進めます
新しいファイルが作れない	エクスプローラー	tpro-work の行を右クリックして「新しいファイル」を選びます

演習中と休憩時間は教室内で声をかけてください。連絡先は当日ご案内します（要確認）。

今日の持ち帰り

身についたこと

できるようになったこと	どこで使うか
文章の仕様を、数えられる受入基準まで書き直す	客先の仕様書からテスト設計に入るとき
受入基準からケース候補をAIに出させ、人が削る	テストケース表を起こす作業
ブラウザだけで走る単体テストを作り、結果画面を見せる	テストツールを入れられない現場
観点を2つに絞ってレビューを頼む	他人のコードに短時間で目を通すとき
落ちた出力を4か所に絞って貼り、原因の見当を3つ持つ	詰まったことを人に相談するとき

明日から使う3手

3日間の中身は、この3手に圧縮できます。

1. 前提を3行書く。使う言語、動かす場所、やってほしくないこと。この3行を先に置くだけで出力の外れ方が減ります
2. 動かして確かめる。読んで納得した時点では、まだ何も確かめていません。ブラウザで開くまでが1セットです
3. 伏せてから貼る。客先名・製品名・案件名・取引先名は業界と工程に丸めます。貼る前に一度目で見る癖をつけてください

手元に残るのは kadai.html と furikaeri_、それと3日間の配布資料です。次に詰まったときは、指示の最初に前提を3行足すところから戻ってきてください。3日間で書いたものは、追加インストールなし、サーバーなしの環境で全部ブラウザで動きました。同じ条件が現場にもあります。

守っていただくこと

演習と提出物では、次の4つを守ってください。

1. 客先名・製品名・プロジェクト名・取引先名は、業界と工程のレベルに丸めます
2. 実データ・実ファイル・スクリーンショットは使いません。配布したダミーデータだけを使います
3. 数値や仕様は具体値を避け、目的・課題・期待効果の粒度で書きます
4. 提出する前に、社外に出ても問題ない内容かをご自身で確認します

> 客先の仕様書やコードをそのまま Copilot に貼らないでください。今日の題材と同じ粒度に丸めた文章へ置き換えるのが、機密の扱い方です。落ちたテストの出力も同じで、全文貼ると端末名・利用者名・社内のフォルダ構成が一緒に流れます。テスト名、期待と実際、止まった場所、添付の有無の4か所に絞ります。

配布フォルダのデータはすべて架空のものです。依頼文そのものを持ち帰るのは問題ありませんが、送信先の環境が客先の規定で許可されているかを確認してから使ってください。