

ClaudeとCursorを使ったコード生成～実装・テストまで

非エンジニアのための業務効率化入門

よくある困りごと

- 勤怠データの手動転記・集計作業
- 問い合わせ記録の管理と整理
- 定型業務のミス防止のためのダブルチェック
- データ入力・転記作業の繰り返し

今日学ぶ意義

- 少しの工夫で業務がぐっとラクになる
- AIツールを使って難しいコードを書かなくても開発できる
- 自分の業務に合わせた改善方法が見つかる



手作業からの解放

今日の学びの流れ

STEP 1
要件定義

STEP 2
Claude/Cursorで実装

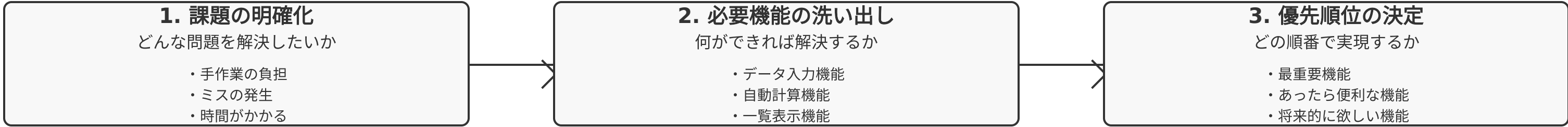
STEP 3
テストと改善

非エンジニアでも安心して取り組める内容です

課題設定と要件定義

まずは解決したい課題を明確にして、必要な機能を洗い出しましょう

要件定義の流れ



実例：手作業での勤怠管理

⚠️ こんな課題はありませんか？

• 勤怠表を紙で記入し、Excel に手入力している

• 集計作業に時間がかかり、ミスも発生する

• データの検索や過去履歴の参照が大変



簡易要件定義表の例

項目	内容
課題	勤怠データの手入力と集計作業の負担が大きい
目的	勤怠管理の自動化で作業時間削減とミス防止
必要機能 （優先順）	1. 勤怠データ入力フォーム 2. 勤務時間の自動計算 3. 月次集計レポート 4. データ検索機能
利用者	従業員（データ入力）、管理者（集計・確認）
実現方法	Claude でコード生成、Cursor で実装・テスト

※要件定義は「何を」実現するかを明確にするものです。「どのように」は次の設計段階で検討します。

💡 非エンジニアでも書ける簡潔な要件定義が重要です

Claudeでコード生成の準備

AIを活用して自然言語からコードを生成する基本的な流れを学びましょう

Claudeへのプロンプト作成

効果的なプロンプトの書き方

- 具体的な課題と目的を明確に伝える
- 必要な機能を箇条書きで列挙する
- 技術的な制約があれば伝える（初心者向け、特定の言語など）
- 出力形式を指定する（HTMLとJavaScriptなど）

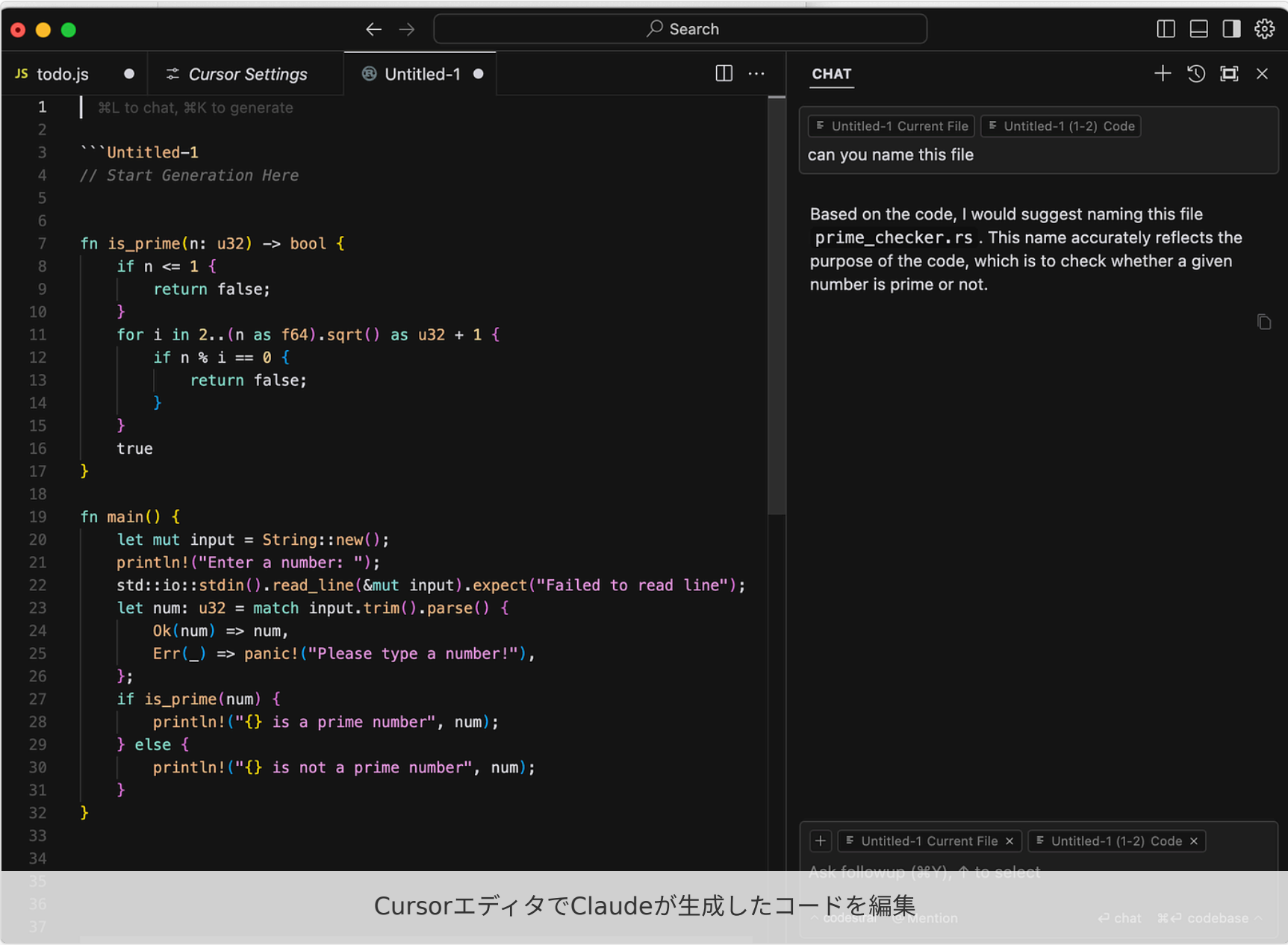


Claudeインターフェースでプロンプトを入力

Cursorのセットアップと使い方

Cursorの基本

- VSCode ベースのAI搭載コードエディタ
- 公式サイトからダウンロード・インストールするだけで簡単に使える
- AIによるコード補完・解説・エラー修正機能を搭載
- 初心者でも扱いやすいインターフェース



CursorエディタでClaudeが生成したコードを編集

実務に近いプロンプト例

💬 勤怠入力画面のプロンプト例：

HTMLとJavaScriptで勤怠入力フォームを作成してください。以下の機能が必要です：

1. 名前、日付、出勤時間、退勤時間を入力するフォーム
2. 勤務時間を自動計算する機能
3. 入力データをテーブルに表示する機能
4. シンプルで初心者にもわかりやすいコード

非エンジニアが理解できるよう、コードにはわかりやすいコメントを付けてください。

💡 プロンプトは具体的に、かつ初心者視点を伝えるのがポイントです

Cursorでのコード生成と実装

生成されたコードを実行し、アプリとして形にしてい 過程を学びます

コードをCursorで実行する

index.html

```
<!DOCTYPE html>
<html>
<head>
  <title>勤怠管理アプリ</title>
  <style>
    body { font-family: Arial, sans-serif; }
    .container { width: 80%; margin: 0 auto; }
    /* 入力フォームのスタイル */
    .form-group { margin-bottom: 15px; }
    label { display: block; margin-bottom: 5px; }
    input { width: 100%; padding: 8px; box-sizing: border-box; }
    /* ボタンのスタイル */
    button { background-color: #4CAF50; color: white; padding: 10px 15px; border: none; cursor: pointer; }
    button:hover { background-color: #45a049; }
  </style>
</head>
<body>
  <div class="container">
    <h1>勤怠入力</h1>
    <div class="form-group">
      <label for="date">日付</label>
      <input type="date" id="date" required>
    </div>
    <div class="form-group">
      <label for="startTime">開始時間</label>
      <input type="time" id="startTime" required>
    </div>
    <div class="form-group">
      <label for="endTime">終了時間</label>
      <input type="time" id="endTime" required>
    </div>
    <button onclick="saveAttendance()">登録</button>
  </div>
```

ポイント

- Claudeが生成したコードをそのままCursorに貼り付けられます
- 上記は勤怠管理アプリの**HTMLとCSS**部分です
- ハイライト部分がフォームとボタンの実装です
- 複雑なコードも自然言語での指示で生成できます

💡 小さな単位で動作確認しながら進めると、問題の早期発見につながります

アプリの実行と確認

勤怠入力

日付

2025-08-15

開始時間

09:00

終了時間

18:00

登録

データが登録されました！

勤怠一覧

日付	開始	終了	労働時間
2025-08-15	09:00	18:00	8.0時間
2025-08-14	08:45	17:30	7.75時間
2025-08-13	09:15	18:30	8.25時間

今月の合計：23.0時間

📄 CursorでLive Previewを使うと、コードを編集しながらリアルタイムで結果を確認できます

エラーが発生した場合の対処法

✖ エラーのあるコード

```
function calculateWorkHours() {
  const startTime = document.getElementById('startTime').value;
  const endTime = document.getElementById('endTime').value;

  // 時間を数値に変換
  const startHours = parseInt(startTime.split(':')[0]);
  const startMinutes = parseInt(startTime.split(':')[1]);
  const endHours = parseInt(endTime.split(':')[0]);
  const endMinutes = parseInt(endTime.split(':')[1]);

  // 総労働時間を計算（分単位）
  let workMinutes = (endHours * 60 + endMinutes) -
    (startHours * 60 + startMinutes);

  // 時間に変換
  const workHours = workMinutes / 60;

  return workHours;
}
```

エラー： 開始時間が終了時間より後の場合、負の値が返されます

✔ 修正版コード

```
function calculateWorkHours() {
  const startTime = document.getElementById('startTime').value;
  const endTime = document.getElementById('endTime').value;

  // 時間を数値に変換
  const startHours = parseInt(startTime.split(':')[0]);
  const startMinutes = parseInt(startTime.split(':')[1]);
  const endHours = parseInt(endTime.split(':')[0]);
  const endMinutes = parseInt(endTime.split(':')[1]);

  // 総労働時間を計算（分単位）
  let workMinutes = (endHours * 60 + endMinutes) -
    (startHours * 60 + startMinutes);

  // エラーチェック：負の値になる場合
  if (workMinutes < 0) {
    alert('終了時間は開始時間より後である必要があります');
    return 0;
  }

  // 時間に変換（小数点以下2桁に丸める）
  const workHours = Math.round(workMinutes / 60 * 100) / 100;

  return workHours;
}
```

解決： エラーチェックを追加し、時間計算を改善しました

エラー対処の手順

1. エラーの特定

ブラウザのコンソールやCursorのエラー表示を確認

2. Claudeに再依頼

エラー内容を具体的に伝え、修正案を依頼

3. 修正と再テスト

提案された修正を適用して動作確認

テストと改善の進め方

作成したアプリを試し、改善を重ねることで完成度を高めます

テスト観点のチェックリスト

📌 正しい入力で正しく動作するか

- ✓

正しいデータを入力してエラーが出ないか
例：勤務開始・終了時刻を入力して保存できるか
- ✓

計算結果は正しいか
例：勤務時間が正確に計算されているか
- ✓

登録したデータが正しく表示されるか
例：入力した内容がリストに反映されるか

⚠️ 異常な入力への対応

- ✗

未入力のまま送信するとどうなるか
例：エラーメッセージが表示されるか
- ✗

誤った形式のデータを入れるとどうなるか
例：文字を入力すべき箇所に数値を入力した場合
- ?

想定外の操作をしたらどうなるか
例：同時に複数の機能を使った場合

💡

テストのコツ

最初から完璧を目指さず、小さな単位でテストと修正を繰り返しましょう。問題を見つけたら、すぐにClaudeに相談して修正案を考えてみましょう。

改善サイクルを回す



まとめ・応用例

今日学んだ内容の要点と、実務で活用できる応用例をご紹介します

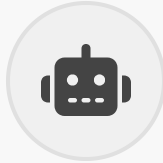
今日の学び：3つのポイント



課題を明確にする

解決したい業務課題を整理し、必要な機能を洗い出す

- ・何が問題なのかを具体的に
- ・どんな機能があれば解決するか
- ・要件定義シートの活用



AIを上手に活用する

ClaudeとCursorを使って効率的に開発する

- ・具体的なプロンプトの書き方
- ・コードの理解と修正の方法
- ・エラー対応のコツ



小さく作って改善する

テストと改善を繰り返し完成度を高める

- ・最小限の機能から始める
- ・テストとフィードバックの重要性
- ・段階的な機能追加

こんな業務に応用できます



問い合わせ管理

顧客からの問い合わせ内容を記録・検索・分類できるアプリ

主な機能： 入力フォーム、検索、タグ付け、集計



在庫リスト生成

商品の入出荷を記録し、現在の在庫状況を一覧表示するアプリ

主な機能： 入出荷記録、在庫表示、アラート通知



自動集計ツール

日々の売上や業務データを入力し、月次レポートを自動生成

主な機能： データ入力、自動計算、グラフ表示



業務フロー管理

チームでの作業進捗を可視化し、ステータス管理を効率化

主な機能： タスク登録、進捗管理、担当者表示

 **ポイント：** 小さな成功体験を積み重ねることで、業務改善の好循環が生まれます

まずは日常業務の小さな「困った」から始めてみましょう

ワーク：自分の業務でアイデアを考えよう

今日学んだ手法を自分の業務に応用してみましょう

✎ 自習課題

自分の業務で役立ちそうなアプリのアイデアを1つ考え、下記の要件定義シートにまとめてみましょう。

📖 ワーク1：業務課題を書き出す

あなたの業務における「困りごと」や「効率化したい作業」は何ですか？

- 現状：どんな手作業が発生している？
- 理想：どうなれば業務が楽になる？

🧩 ワーク2：必要な機能を3つ書く

アプリに実装すべき重要な機能を挙げてみましょう。

- 必須機能：これだけは欲しい機能
- 優先順位：どの順で実装するか

🎵 ワーク3：Claudeへの依頼文を作る

Claudeに依頼するプロンプトを書いてみましょう。

- 解決したい課題を明確に伝える
- 必要な機能や制約を箇条書きで

簡易要件定義シート

※ このテンプレートをノートやテキストエディタにコピーして使用してください

アプリ名	(例：部署別勤怠管理アプリ)
解決したい課題	(例：紙ベースの勤怠記録の非効率さとミスが多い点)
目的・ゴール	(例：勤怠記録の電子化と集計作業の自動化)
必要機能 (優先順)	1. (例：勤務開始・終了記録機能) 2. (例：部署別集計機能) 3. (例：月次レポート出力機能)
利用者	(例：一般社員、管理職、総務部)
実装方法	(例：HTML/JavaScript、データはLocalStorage保存)

★ あなたにもできます！