コーディングエージェントにフロントエンドを書かせるとき、フィードバックはどのように与えるか？

なぜフロントエンド開発におけるフィードバックは難しいのか

- コードの実行環境が異なる
  - フロントエンドの場合はブラウザで実際に実行するまで正しさを検証できないのにも関わらず、エージェントが直接ブラウザにアクセスする手段を持たない。目隠しをしてプログラミングしているようなもの。そのため、不完全な実装のまま「実装が完璧に完了しました！」と報告することがよくある

- デザインの評価問題
  - デザインには明確な正解がなく、適切なフィードバックを与えづらい
  - 構文上誤りがある CSS を書いてしまったとしても、その誤りに対するフィードバックを受ける機会がない
  - 「白い背景に紫のグラデーション」のような画一的なデザインを生成しがち

このポスターでは上記の問題についてどのような手法で解決が試みられているのか、以下の 4 つについて紹介します。

ブラウザを操作する MCP ツールを使う

Playwright、Chrome DevTools、Claude in Chrome などのツールを使い、ブラウザ操作を自動化して AI に自律的にコードの変更の妥当性を検証してもらいます。クリックやナビゲーションのような操作を行い、スクリーンショットを撮影・アクセシビリティツリーを取得・開発者ツールにアクセスなどの方法でフィードバックを得る。

- エージェントがブラウザを操作するので自動化できる
- DOM ツリーやアクセシビリティツリーを取得できるので、構造化された情報を取得できる
- スクリーンショットはしばしば情報が冗長であり、エージェントが必要な情報を正確に把握できない場合がある
- MCP ツール呼び出しのたびに中間結果がコンテキストに追加され、コンテキストを圧迫しやすい

人間がブラウザを操作し、AI に適切なコンテキストを与えるツールを使う

この手のツールには React Grab、Cursor Browser があります。
React Grab はブラウザ上で要素を選択すると、対応するコードコンテキスト（ファイルパス、行番号、コンポーネント）を自動抽出してエージェントに提供するツールです。Cursor Browser はドラッグで要素移動やスタイル変更の結果を指示として渡せます。

- 人間による柔軟なブラウザ操作と、正確かつ無駄がないコンテキストの提供を両立できる
- アドホックなフィードバックには最適だが、自動化操作には適していない。

Figma の信頼できる情報源からデザインデータを取得する

Figma MCP を使用して信頼できる情報源であるデザインデータからデザイントークン・コンポーネントの情報を取得しそれを元にデザインを行ってもらいます。

- プロダクトのデザインに一貫した実装が可能になる（画一的なデザインの問題に対する解決策）
- Figma の側のファイル設計の質に大きく左右される
- 実装したデザインに対するフィードバックは別途与える必要がある

Skills を使用してフロントエンドのデザインガイドラインを提供する

専門知識を提供する Skills によってデザインの実装時のガイドラインを提供する。Anthropic が提供する Frontend Design Plugin により成果物の品質が向上したことが語られています。プロジェクト固有のデザインガイドラインがあれば、Skills を通じて伝えるのが良いでしょう。
ただし現状の Skills の仕組みでは、エージェントが自律的に Skill を使ってくれる確率が低いです。「Design Skill を参照して」のように明示的に指示する必要がある点に注意してください。

結論：目的に応じて手法を組み合わせる

- プロダクトのデザインに一貫した実装が可能になる（画一的なデザインの問題に対する解決策）
- Figma の側のファイル設計の質に大きく左右される
- 実装したデザインに対するフィードバックは別途与える必要がある

azukiazusa

Web フロントエンドが好きな Web アプリケーションエンジニア