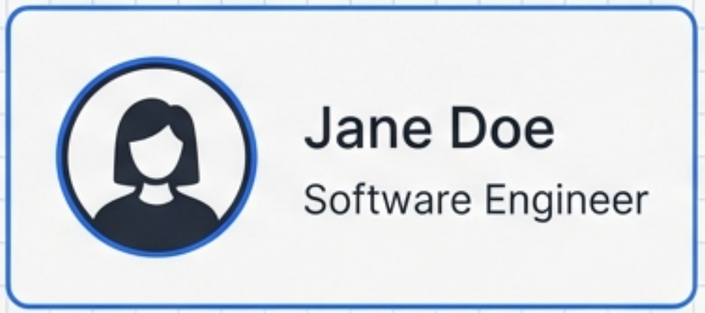
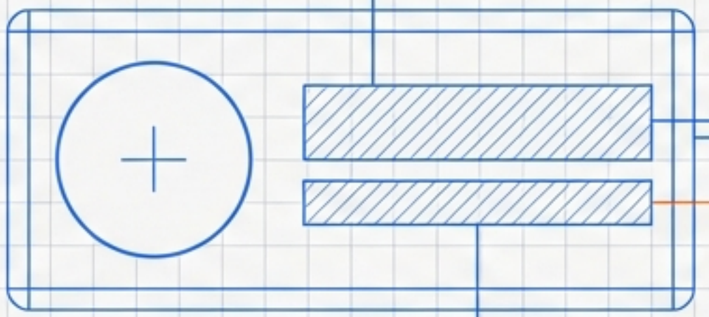
Pencil: UIデザインと実装の境界を溶かす革命

Claude Code & MCPが切り拓く「デザイン・アズ・コード」の未来



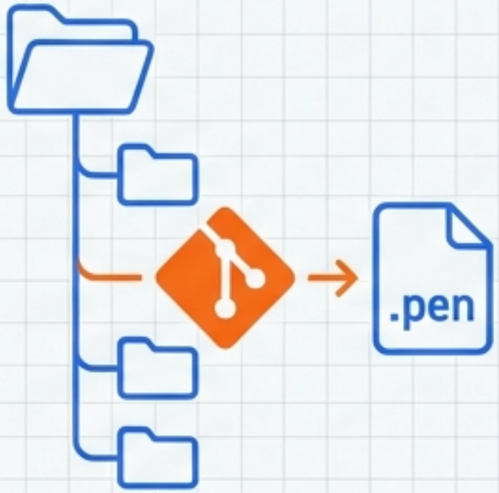


```tsx
import React from 'react';
import { Card, Avatar, Text } from '@pencil/ui';

const ProfileCard = ({ user }) => {
  return (
    <Card className="p-4 rounded-lg border border-blu
      <div className="flex items-center">
        <Avatar src={user.avatarUrl} alt={user.name}
        <div className="ml-4">
          <Text className="text-lg font-bold text-cha
          <Text className="text-sm text-charcoal-ligh
        </div>
      </div>
    </Card>
  );
};

export default ProfileCard;
```

Vibe Designing Has Arrived. // v1.0

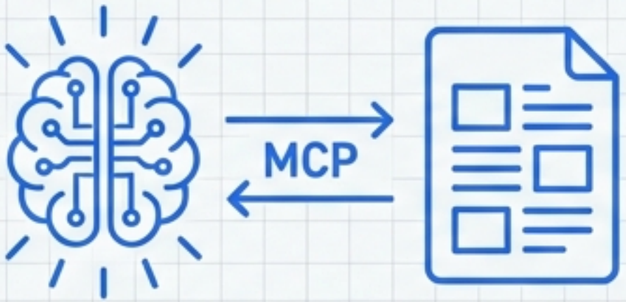
リポジトリに住む「AIネイティブ」デザインキャンバス

Pencilは、開発環境（VS Code/Cursor）とGitワークフローの中に、無限のデザインキャンバスを統合します。



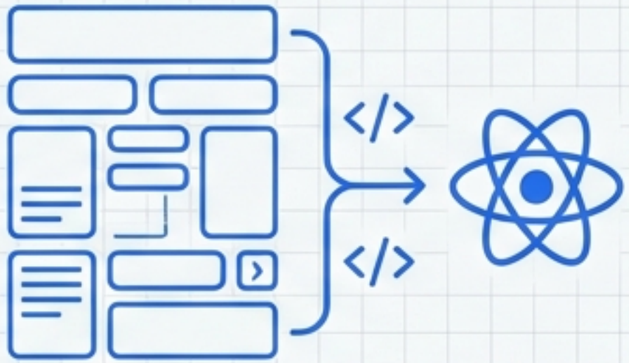
Git-Native Assets

デザインファイル（.pen）はJSON形式でリポジトリ内に保存。コードと同じバージョン管理下で、プルリクエストによるレビューが可能。



Agent-Driven (MCP)

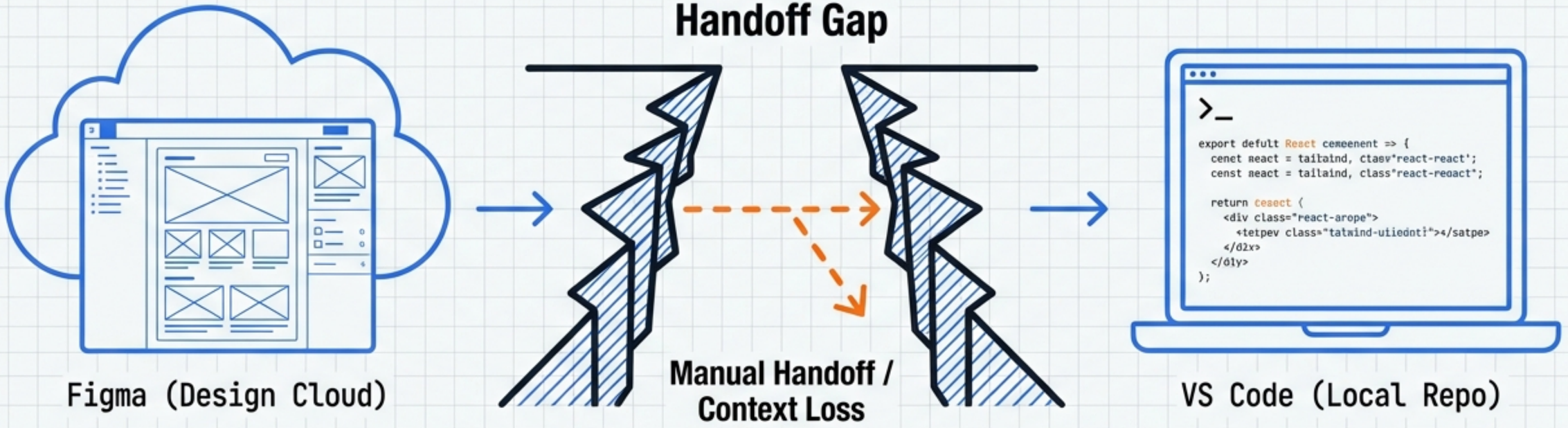
画像認識（Vision）ではなく、MCPを通じて「構造データ」としてAI（Claude Code等）と双方向に同期。



Production-Ready Output

画像生成ではなく、React, Tailwind CSS, shadcn/uiなどの実用的なコンポーネントコードを出力。

従来の課題：フィグマとコードの間の「深い溝」



Synchronization Drift (乖離)

実装が進むにつれ、デザインデータと実際のコードが即座に同期しなくなる。「Figmaは絵、コードが正」という状態への諦め。

Context Loss for AI (コンテキストの欠如)

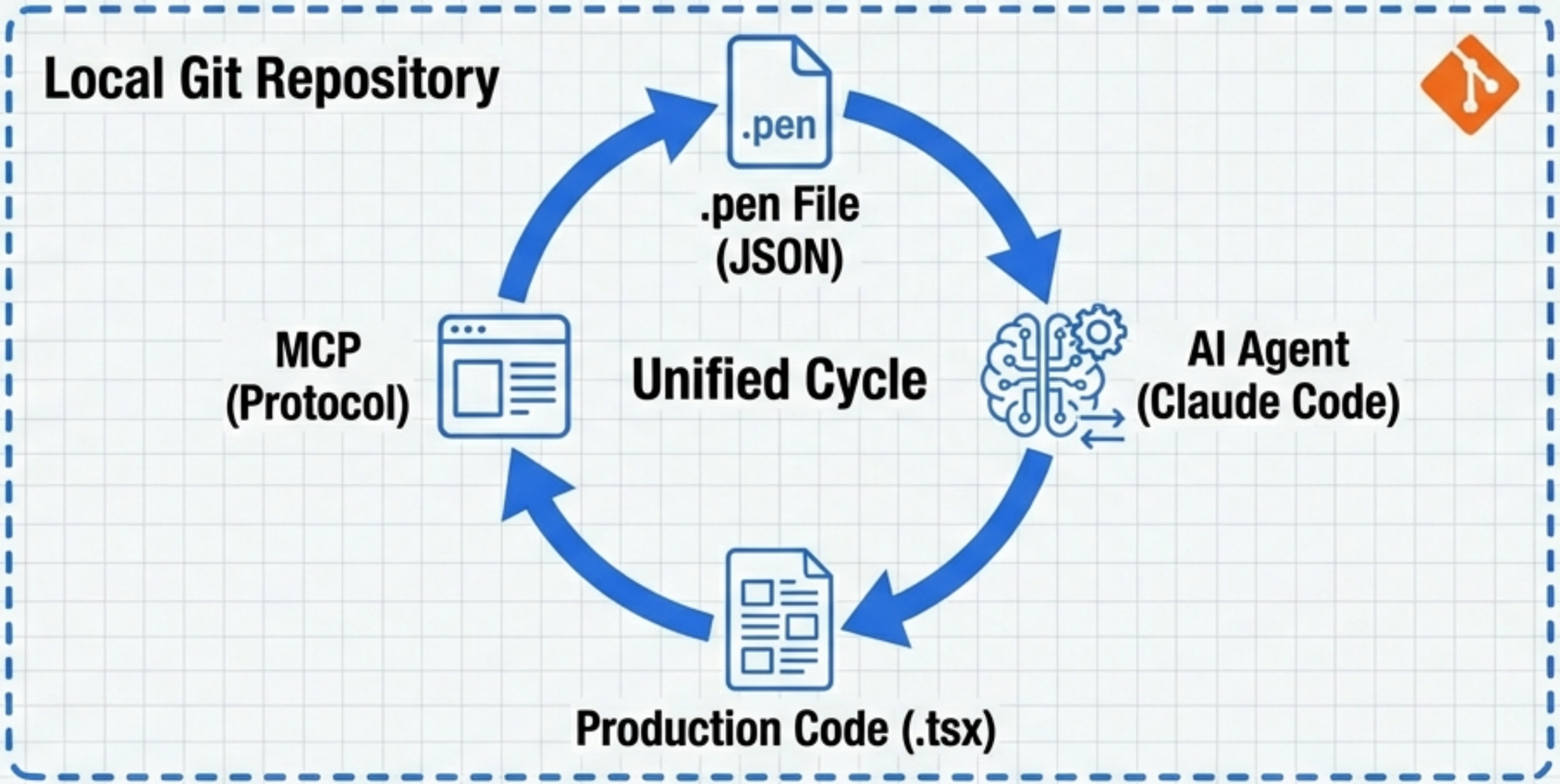
従来のAIエージェントはスクリーンショット（ピクセル）しか見ることができず、コンポーネントの構造やProps、プロジェクトのルールを理解できない。

Siloed Assets (資産の分断)

デザインがGit管理外にあるため、エンジニアリングの資産として扱われない。

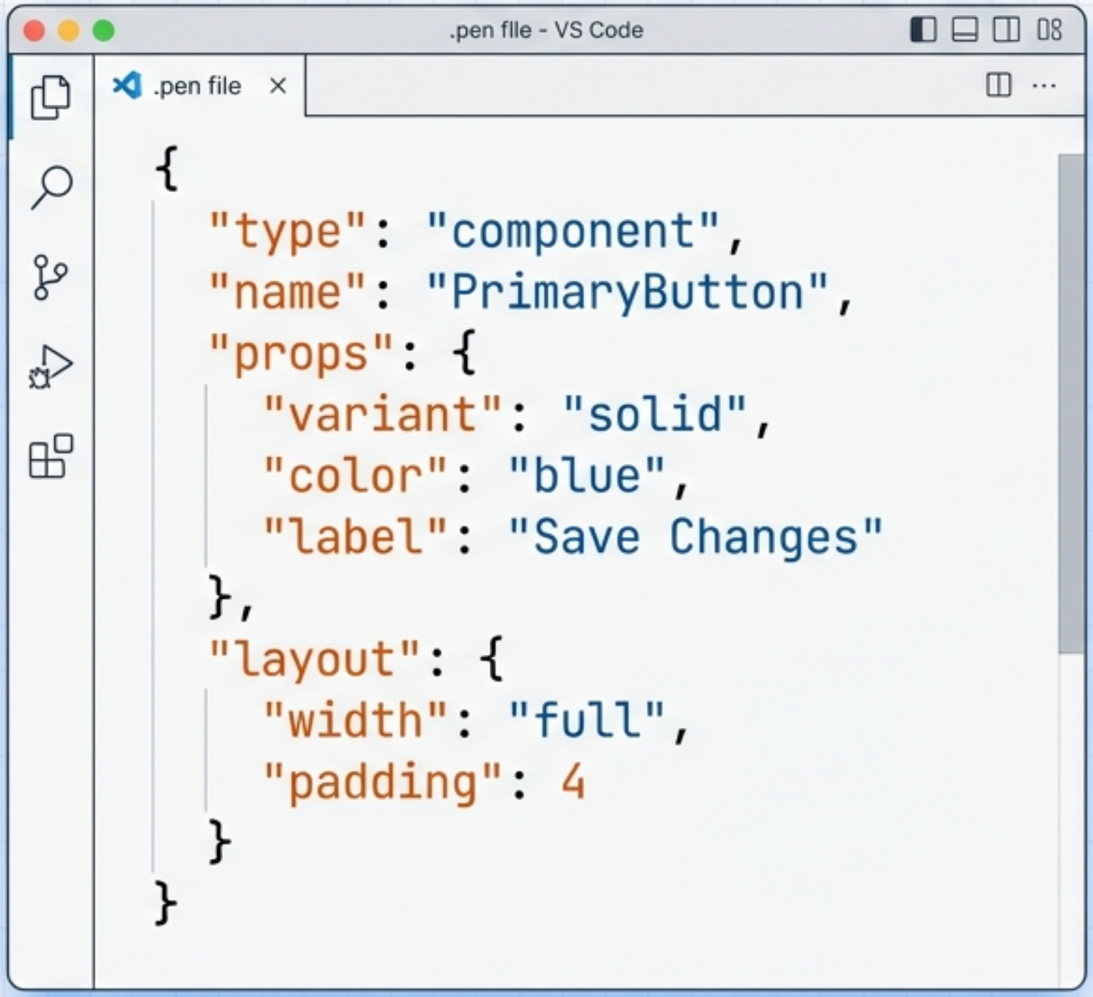
解決策：デザインを「エンジニアリング資産」へ格上げする

Pencilはデザインを「絵」ではなく、リポジトリ内の「構造化データ」として扱います。



- Handoff Zero: エンジニア自身がリポジトリ内でUIを調整し、そのままコミット。
- Single Source of Truth: デザインとコードが物理的に同じ場所に存在し、相互に参照可能。

技術仕様：.pen ファイルとGit統合の威力



- **Version Control (バージョン管理)**
  コードの変更（機能追加）とデザインの変更（UI調整）を、1つのコミットに含めることが可能。
- **Diff & Merge (差分確認)**
  テキストベースであるため、GitHub上でUI変更の差分（Diff）を確認し、プルリクエストで議論できる。
- **No Lock-in (ロックインなし)**
  データはローカルにあり、オープンなフォーマット。クラウドサービスへの依存度を下げる。

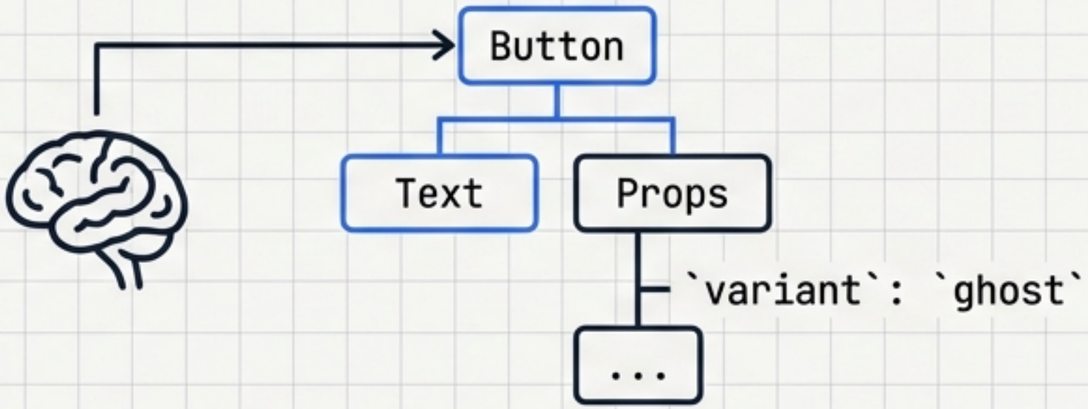
エンジン：MCP (Model Context Protocol) による構造的理解

Vision vs. MCP

Vision (Traditional AI)

AI looks at pixels.
Prone to hallucinations on padding/margins.

MCP (Pencil Engine)



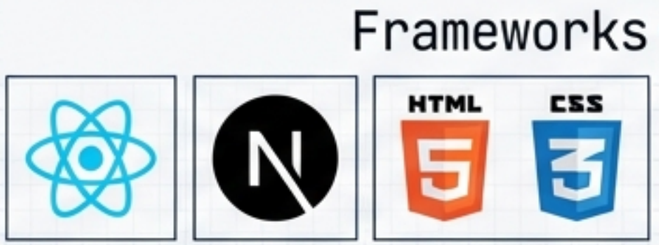
AI reads the DOM Structure.
Precise manipulation of properties.

Update `Button` component `variant` to `ghost`.

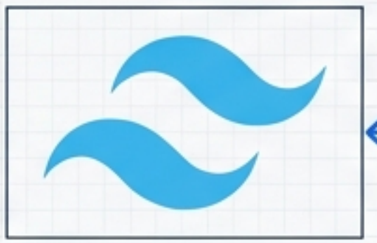
The Result: AIエージェントは「絵を描く」のではなく、「仕様に基づいてDOMノードを操作」する。　　これにより、高精度のバイブコーディング（Vibe Coding）が実現する。

実装スタックへの最適化：プロトタイプで終わらせない

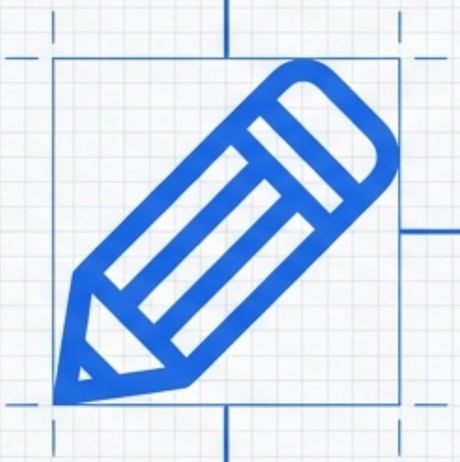
Whatever your stack uses.（あなたのスタックに合わせて生成）



Component-based architecture.



Best practice utility classes.

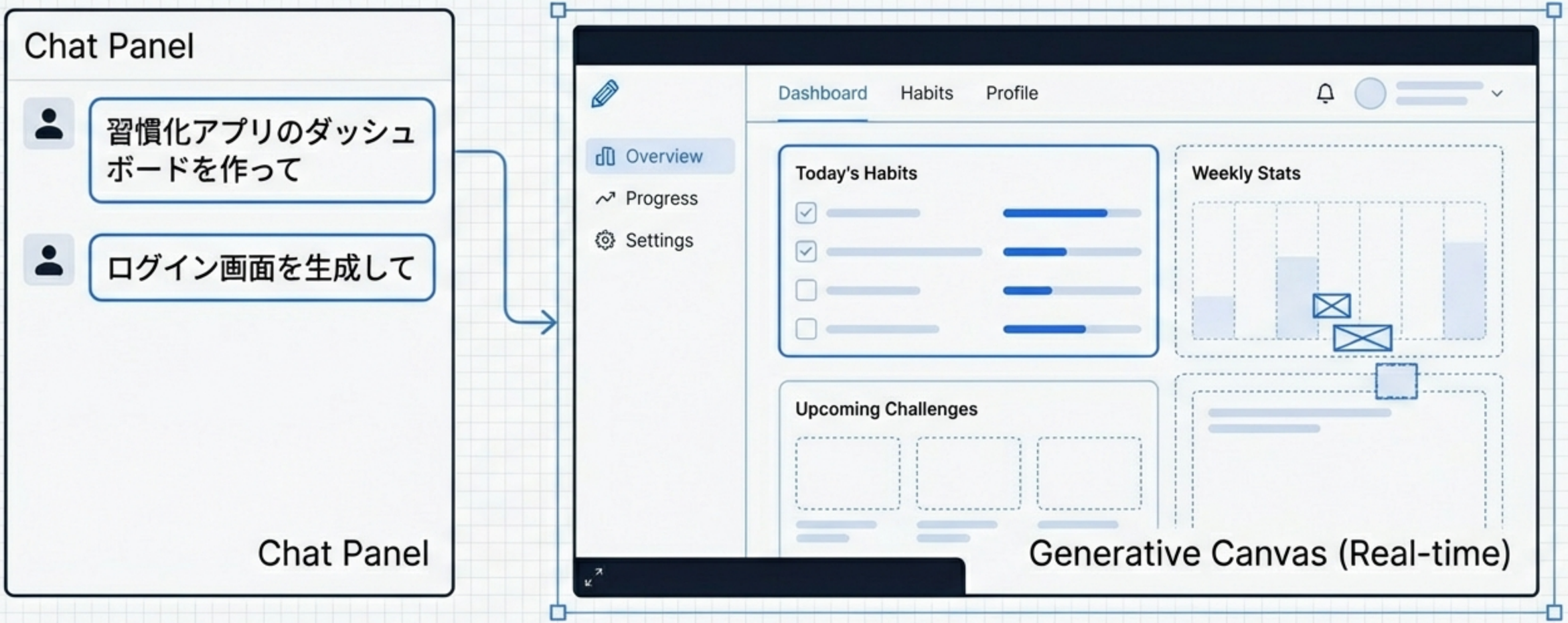


Lucide Icons

shadcn/ui をネイティブサポート。
AIは使い捨てのコードではなく、
shadcnのコンポーネント構造を前提とした
コードを生成する。

Flexibility: プロジェクト固有のルールや、他のUIパッケージ（Halo, Lunarus等）もコンテキストとしてAIに渡すことで対応可能。

Workflow Step 1: 対話型デザイン生成

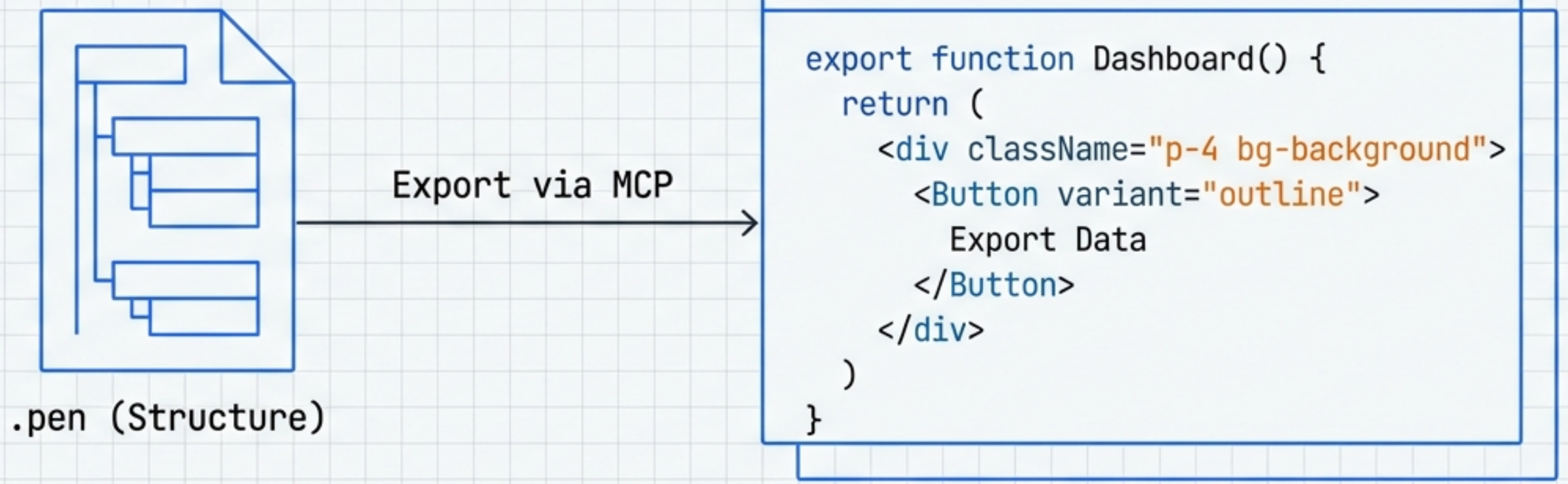


[1] **Prompting**: 自然言語で指示。

[2] **Generative Canvas**: AIがリアルタイムで要素を配置。

[3] **Bi-directional Editing**: 人間が手動で微調整することも、AIに追加指示を出すことも可能。

Workflow Step 2: 構造データからコードへの変換



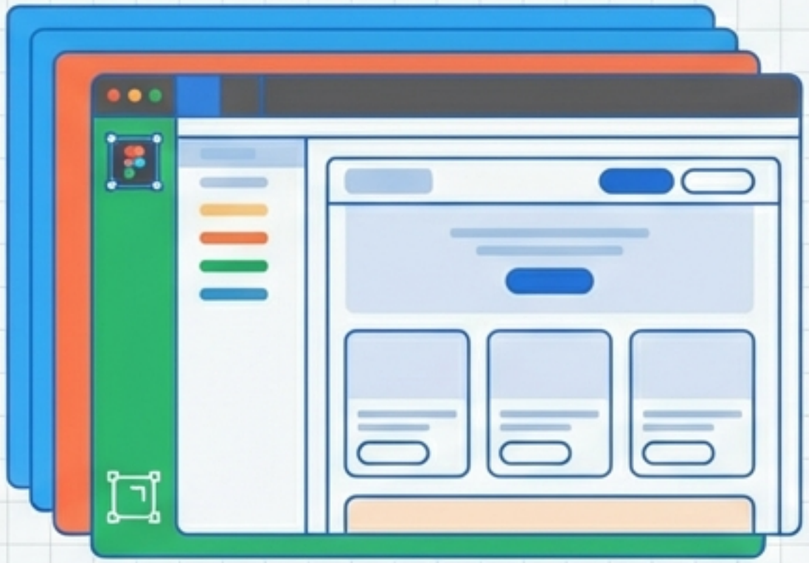
Context Injection: Claude Codeなどのエージェントが、.penファイルのDOM構造を読み取る。

Coding: 指定されたスタック（React + Tailwind）に合わせて.tsxファイルを書き出す。

Quality Assurance: スパゲッティコードではなく、コンポーネントを再利用した記述が可能。

Workflow Step 3: Figmaからのインポートと移行

Bridging the Legacy



Figma Interface

Copy & Paste

Preserves layers, styles, and layout.

Pencil Interface (Git-native)

Scenario: 既存のFigma資産がある場合。

Solution: Figmaのレイヤー構造を保持したままPencilにペースト可能。レガシーなデザインファイルをGit 管理下のAI ワークフロー（Pencil）に持ち込む「移行パス」として機能。

ガバナンス戦略：破綻しないための運用ルール

1. Component Parity (コンポーネント等価性)

Pencil上のコンポーネント（Props/Variant）を、React実装側のPropsと完全に一致させる。
「実装にないVariant」をデザインで作らない。

2. Semantic Tokens (意味論的トークン)

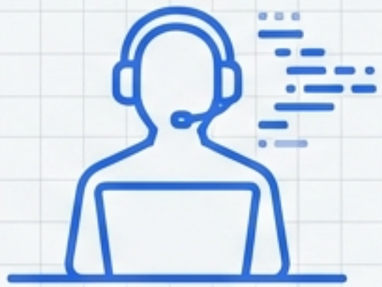
色や余白は `hex` や `px` で直書きせず、`bg-surface` や `spacing-4` などのトークン (Tailwindクラス) を使用する。

Do / Don't	
Don't ✗	**Do** ✓
`background: #ffffff`	`className="bg-surface"`

3. Define States (状態の定義)

Happy Path（通常時）だけでなく、Loading, Empty, Error, Validationの状態をPencil上で定義する。

誰のためのツールか？ (Target Audience)



Solo Developers / Indie Hackers

デザイナー不在でも、フルスタックな開発スピードを最大化したい個人。

Engineer-Designers

コードの構造でデザインを考ええる、エンジニアリング思考のクリエイター。

Small & Agile Teams

「Handoff（受け渡し）」のボトルネックを解消し、実装までのリードタイムをゼロにしたいチーム。

Not（Yet）For: 厳密なピクセルパーフェクト管理と大規模な分業が必要なエンタープライズデザイン組織。

コミュニティの反応：「Vibe Design」の幕開け

Claude Code以来の『デザイン革命』

もうFigmaのデザインモードは使わないかもしれない

エンジニアが待ち望んでいたインターフェース

Trend: Vibe Coding → Vibe Designing

現時点での評価（Pros & Cons）

Pros（メリット）	Cons / Challenges（課題）
✅ Complete Free (Currently): 無料で利用可能。 ✅ Git Integration: デザイン資産のバージョン管理と透明性。 ✅ High Fidelity Code: MCPによる高精度なコード生成。	⚠️ Manual Tools: 手動での細かいデザイン調整機能はFigmaに及ばない（発展途上）。 ⚠️ Setup: MCPや拡張機能の導入など、環境構築が必要。 ⚠️ Ecosystem: プラグインやコミュニティリソースはまだ少ない。

デザインと実装の分離は、ここで終わる

Pencil is the new paradigm for managing UI as "Source Code".

1. **VS Code / Cursor 拡張機能をインストール**
2. **プロジェクト内に .pen ファイルを作成**
3. **AIエージェントと共に、最初の 'Vibe Designing' を体験する**

pencil.dev