

AIを使うときの画面切り
替えが嫌で、ユーティリ
ティツールを開発してた
ら、思ったより良くて、
グローバルtoCプロダクト
を開発している話



奥脇 真人

Masato Okuwaki

コレ株式会社 代表取締役

AIプロダクト開発、AIコンサルティング

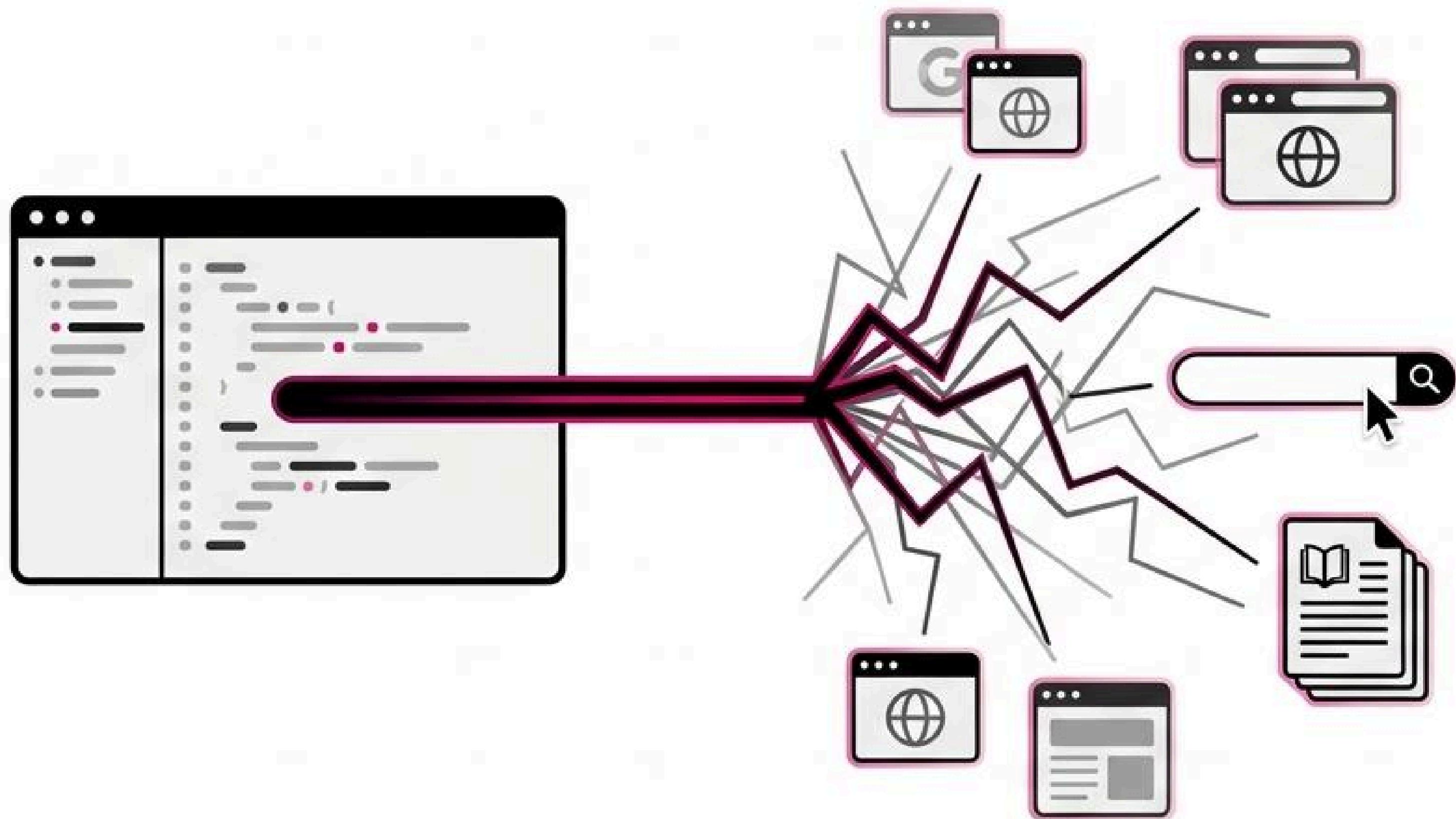
X: @okuwaki_m



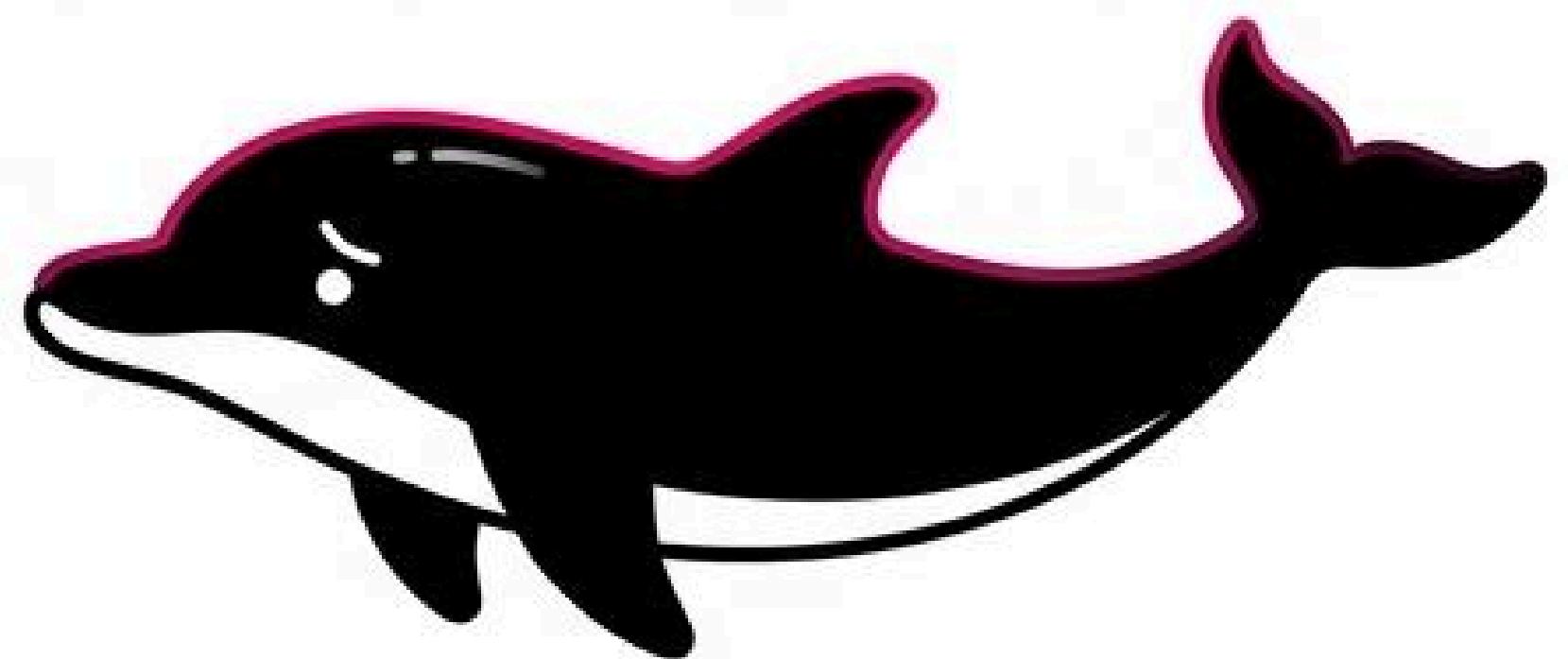
AIツールを使うとき 画面切り替え多いの苦痛



その「ちょっと調べる」が、集中力を奪っていく。



複雑なコード、解読不能なエラー、新しい技術文書…。理解のためにブラウザを開いた瞬間、私たちの集中は霧散します。この「コンテキストスイッチ」のコストは、想像以上に大きいのです。



自分のためだけの開発

macOS

画面常駐

高速

可愛いキャラ

プロンプト打たなくていい

図解も読める

画像もつくれる

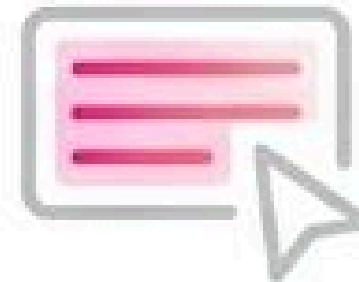
動画もつくれる

ターミナル起動できる

思考を止めない、フルスクリーンAIアシスタント



画面上のあらゆるものを、ショートカット一つで瞬時に理解する。



テキストを選択:



スクリーンショット:



URLを選択:



IrukaDark

画面を変えずに
ショートカットキーをポチッ



Cursorの上で コードを選択してOption+A

```
11 // Prefer explicit identity from env
12 const envName = process.env.CSC_NAME || process.env.MAC_CODESIGN_IDENTITY;
13 if (envName && envName.trim()) return envName.trim();
14
15 // Fallback: derive from the signed main app
16 try {
17   const out = execFileSync('codesign', ['-dv', '--verbose=2', appPath],
18   encoding: 'utf8',
19   stdio: ['ignore', 'pipe', 'pipe'],
20 });
21 // Pick the Developer ID Application authority line
22 const match = out.match(/Authority=Developer ID Application:.*\(.*\)/);
23 if (match && match[0]) {
24   // codesign -s accepts the full authority string
25   return match[0].replace('Authority=', '').trim();
26 }
27 } catch {}  

28 return null;
29
30
31 module.exports = async function (context) {
32   if (context.electronPlatformName !== 'darwin') return;
```

おっ、このコードブロックについて詳しく説明するね。これは、macOSのアプリの署名情報を取得するための処理みたいだ。

重要点：macOSアプリの署名情報を取得している処理だよ

このコードの主な目的は、指定されたアプリケーションのパス（`appPath`）を使って、コードサイン（`codesign`）の情報を取得

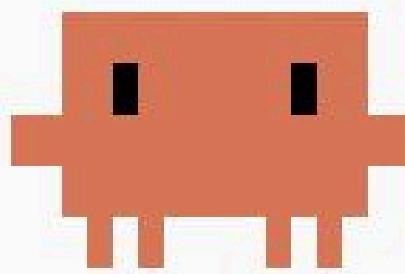
イルカダークに質問する



Claude Codeの出力を選択して Option+A

- 3. [ja/downloads/index.html:84](#) - Apple Silicon版のダウンロードリンクをv1.2.3に更新
- 4. [ja/downloads/index.html:94](#) - Intel版のダウンロードリンクがv1.2.2からv1.2.3に更新されました。

→ [~/Projects/dev_irukadark/irukadark_public/irukadark](#)



Claude Code v2.0.42
Sonnet 4.5 · Claude Max
/.../okuwakimasato/Projects/dev_irukadark

> Try "how do I log an error?"

? for shortcuts

チャット クリップボード スニペット ランチャー

Claude Code v2.0.42
██████████ Sonnet 4.5 · Claude M...

ClaudeのモデルがSonnet 4.5やMaxを使いつつ、特定の開発プロジェクトのディレクトリで何らかの処理が行われている状況だね。

+ Irukadarkに質問



AI技術論文のURLを選択して Option+Q

The screenshot shows a web page for an AI-generated abstract from arXiv. The top navigation bar includes a 'BETA' indicator, the arXiv logo, a message about experimental HTML, and links for 'Why HTML?', 'Report Issue', 'Back to Abstract', 'Download PDF', and a gear icon.

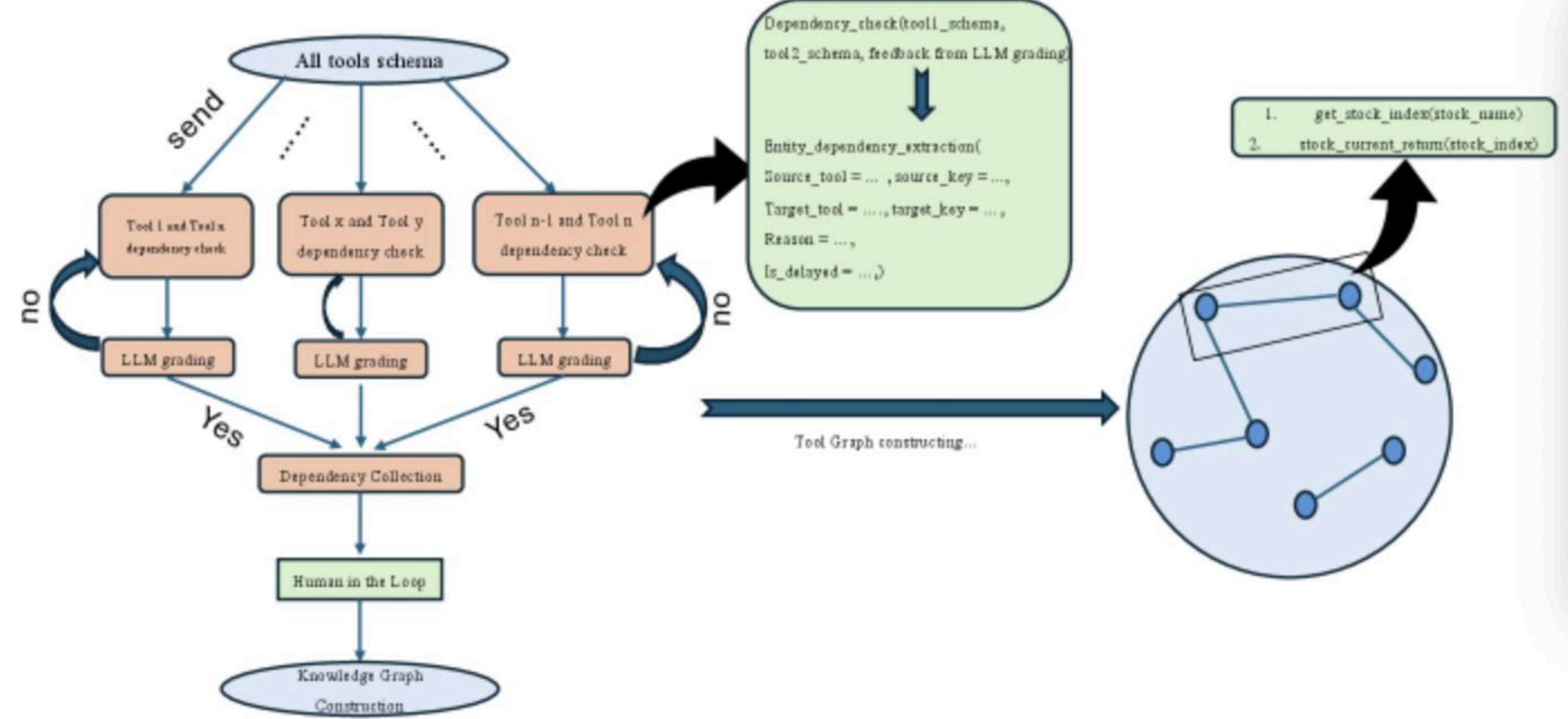
The main content area displays the title 'Bridging Tool Dependencies and Domain Knowledge: A Graph-Based Framework for In-Context Planning', authors (Shengjie Liu, Li Dong, Zhenyu Zhang), their email addresses, and the license information (CC BY 4.0). A sidebar on the right provides a detailed explanation of the '選択範囲の解説' (Selection Range Explanation) for the highlighted text, mentioning the framework's purpose of bridging tool dependencies and domain knowledge through a graph-based approach. It also lists a point of interest related to the tool's knowledge graph construction.

Abstract

We present a framework for uncovering and exploiting dependencies among tools and documents to enhance exemplar artifact generation. Our method begins by constructing a tool knowledge graph from tool schemas—including descriptions, arguments, and output payloads—using a DeepResearch-inspired analysis. In parallel, we derive a complementary knowledge graph from internal documents and SOPs, which is then fused with the tool graph. To generate exemplar plans, we adopt a deep–sparse integration strategy that aligns structural tool dependencies with procedural knowledge. Experiments demonstrate that this unified framework effectively models tool interactions and improves plan generation, underscoring the benefits of linking tool graphs with domain knowledge graphs for tool-augmented reasoning and planning.

1 Introduction

よくわからない図解を選択して Option+S



選択範囲の解説

この図は「ツールグラフの構築（Tool Graph Construction）」のプロセスを示しているよ。

簡単に言うと、複数のツール（Tool 1 から Tool nまで）がお互いにどういう依存関係にあるかを調べて、それをグラフ（Tool Graph）としてまとめる流れなんだ。

イルカダークに質問する

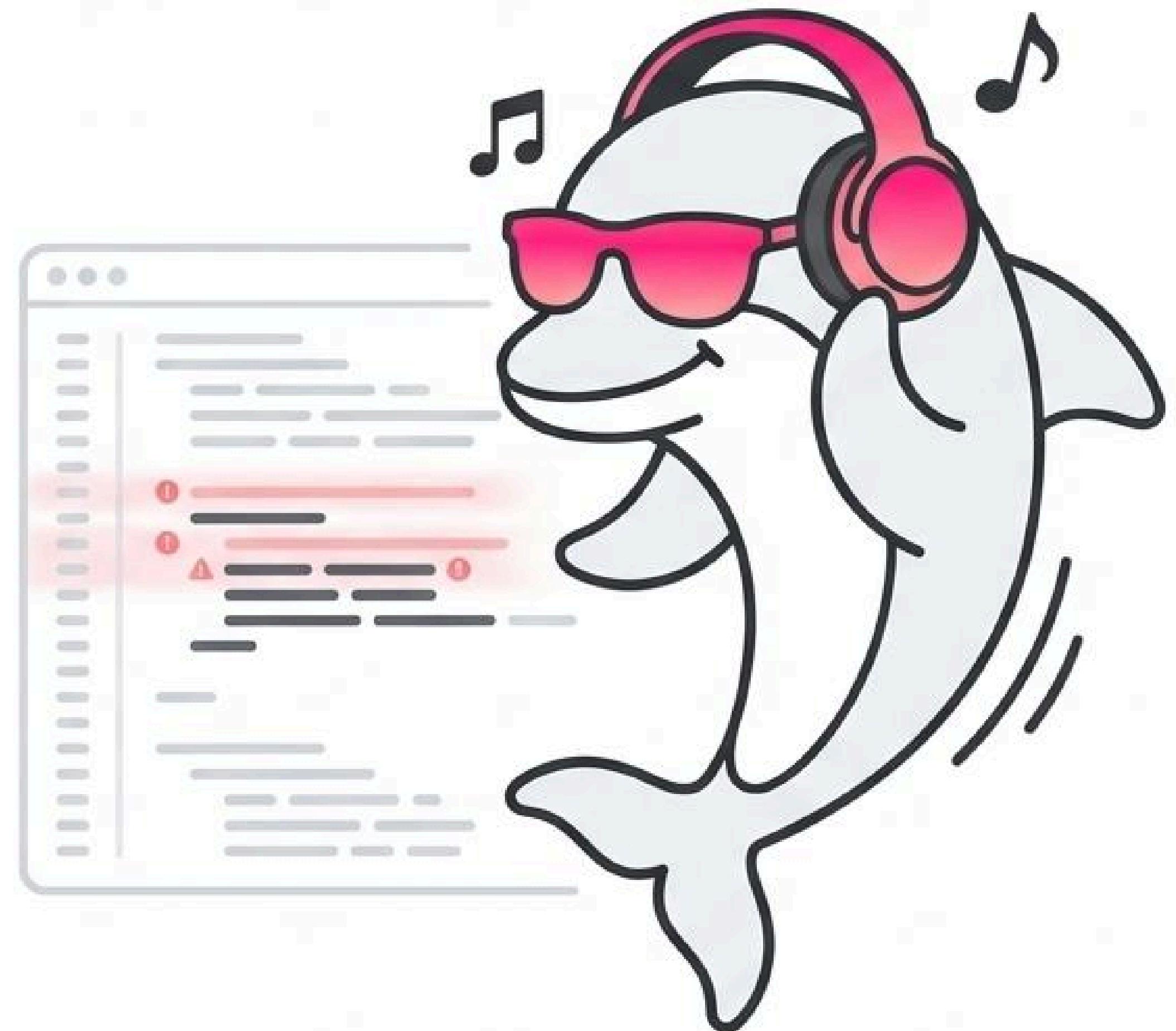


Figure 1: Tool Graph Construction

利用シーン①: Vibe Coding × エラーハンドリング

ノリと感覚でコーディングする人たち。
エラーが出たり、知らないツールを
使ったりする時も、画面を遷移せずそ
の場で解決したい。

「バイブスを途切れさせない」ために
使われている。



利用シーン②：熟練エンジニア × 未知の技術

AI駆動開発により、熟練者でも初めて触る言語や技術に取り組む機会が爆増。RAG、マルチモーダル等の最新論文や英語ドキュメントを必死に読む際に重宝されていた。



デスクトップツール？絶対にやりません。

Clipy, Alfred, Raycast...

市場は成熟した神ツールで溢れている。

ビジネスとして参入するのは正気の沙汰ではない。

レッドオーシャン。



「マーケットイン」は仕事でやる。

個人の開発では真逆に行く。「他人のニーズなんて知ったことか。
僕のパフォーマンスが上がればそれでいい」。



クライアントに見せたら「何それ！？」

自分専用のつもりで誰にも見せる予定はなかった。ある日、クライアント先のエンジニアの前でうっかり使ったら、驚かれた。「あれ、これ他人にも需要ある？」と初めて思った瞬間。



**でもAIツールは日本では
説明工数が大変・・・**

試しに海外向けにX投稿したら 1000件くらいダウンロードされた



いけるかもしない！

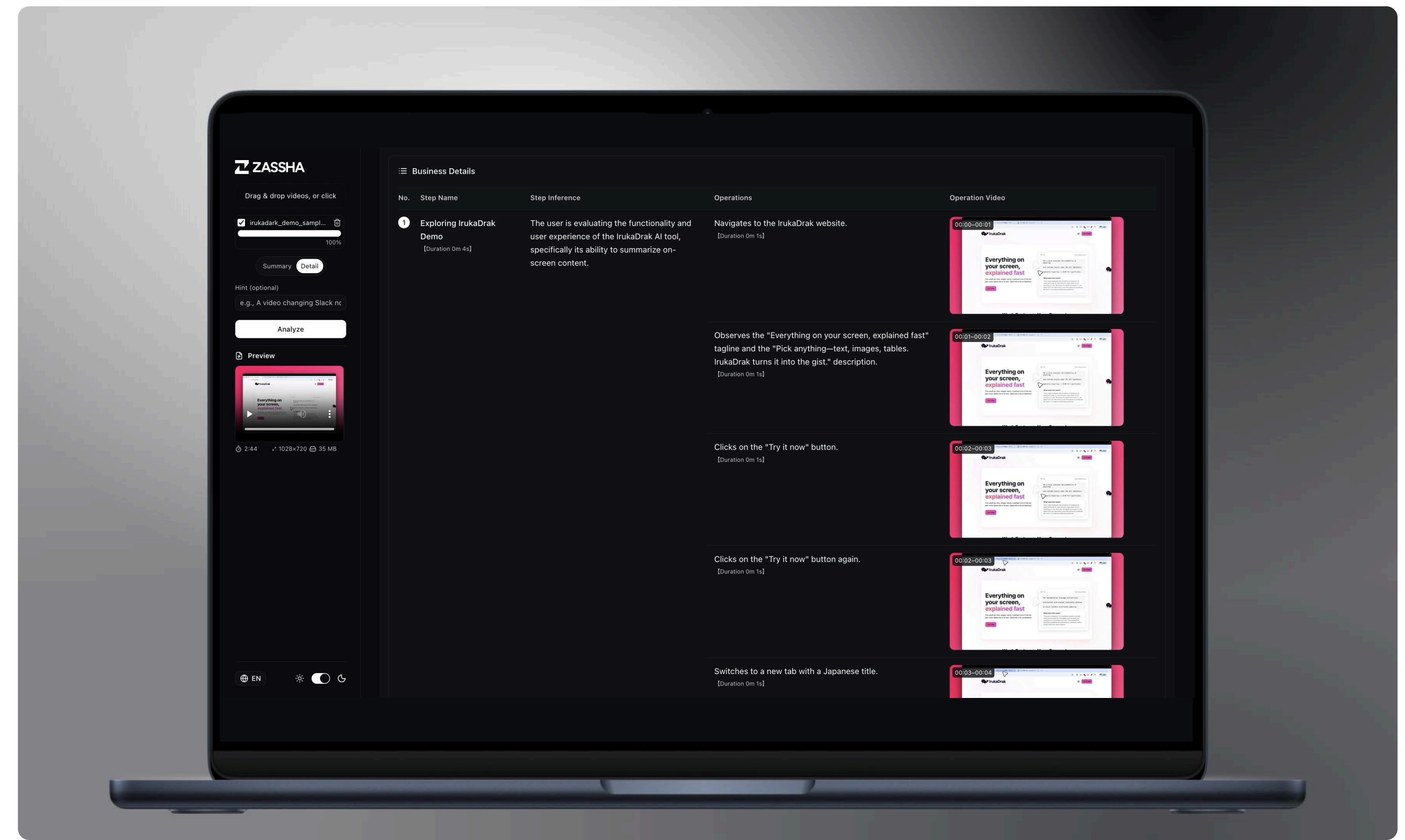


**海外のtoCプロダクト展開を調査
プロダクトハントがいいらしい**

とりあえず一回プロダクトハント出す用に プロダクトをつくった。結果10位...



作業動画をアップロード
すると、動画を分割して
作業マニュアルをつくっ
てくれるAI



いろいろ聞いたり調査したら
しっかり準備するのが大切な事

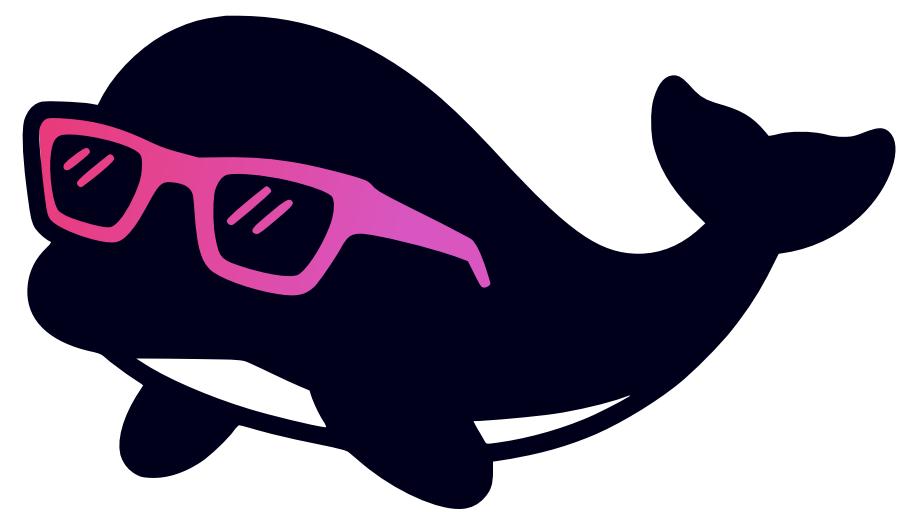
準備って...

AIで作れるものは増えた
しかし、広めるのが難しい
日本からグローバル攻めるのは
孤軍奮闘は難易度が高い

AI駆動開発で個人向けツールを高速でつくり
プロダクトハントで応援しあいましょう



X : @okuwaki_m



IrukaDark

