

今日からフロントエンドの 魔法使い

～魅せる技で創るユーザーが恋するUX～ ❤️

SGE 輪読会の成果発表会 × LT会 2024.12.02

自己紹介

 グンタ ブルンナー
Günther Brunner

 CyberAgent since 2012

-  CTO統括室  > **Developer Productivity室**

 AI  UX  Design  Performance
 Music  Movies  Sushi  Travel

 @gunta85

 @gunta

 dev.to/gunta

 zenn.dev/gunta

 guntherbrunner.art 

 guntherbrunner.art/music/ 

12年間のCAキャリア



メディア部門



- サービス精神の醸成
- UXデザインスキルの向上
- ユーザー中心の思考

横軸部門



- SLO (Service Level Objectives)
- 開発生産性の追求 ➡️ 🚀 (今ここ)
- チーム横断的な改善活動<

ゲーム部門



- フロントエンド技術の極限追求
- パフォーマンス最適化
- インタラクティブ表現の探求

AI部門



- バックエンド技術の習得
- Figmaの習得
- 生成AIの実践的活用
- 最新技術のキャッチアップ



各部門での経験が、現在のフルスタック開発力に繋がっています

前半：ツール編

開発生産性の加速

導入技術の紹介

2024/12

翻訳はもう維持されていません、[英語のドキュメント](#)をご覧ください。



ガイド ▾ テーマ ▾ カスタマイズ ▾ リソース ▾ 日本語 ▾



Search

Sliddevはまだ開発中です。APIや使い方はまだ確定ではありません。



Sliddev

Presentation Slides for Developers Beta

はじめる

もっと詳しく

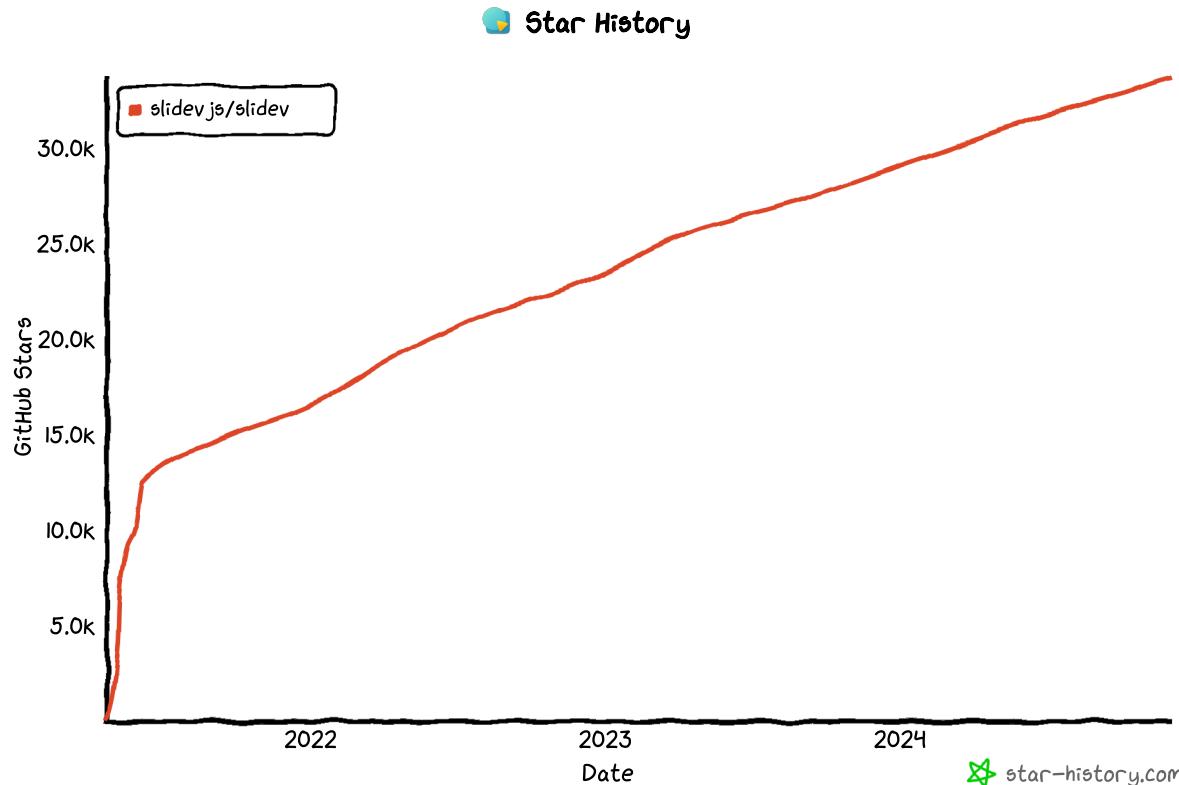
Slidevとは何ですか？

Slidevは開発者向けのスライドメーカー・プレゼンターで、以下の機能がある

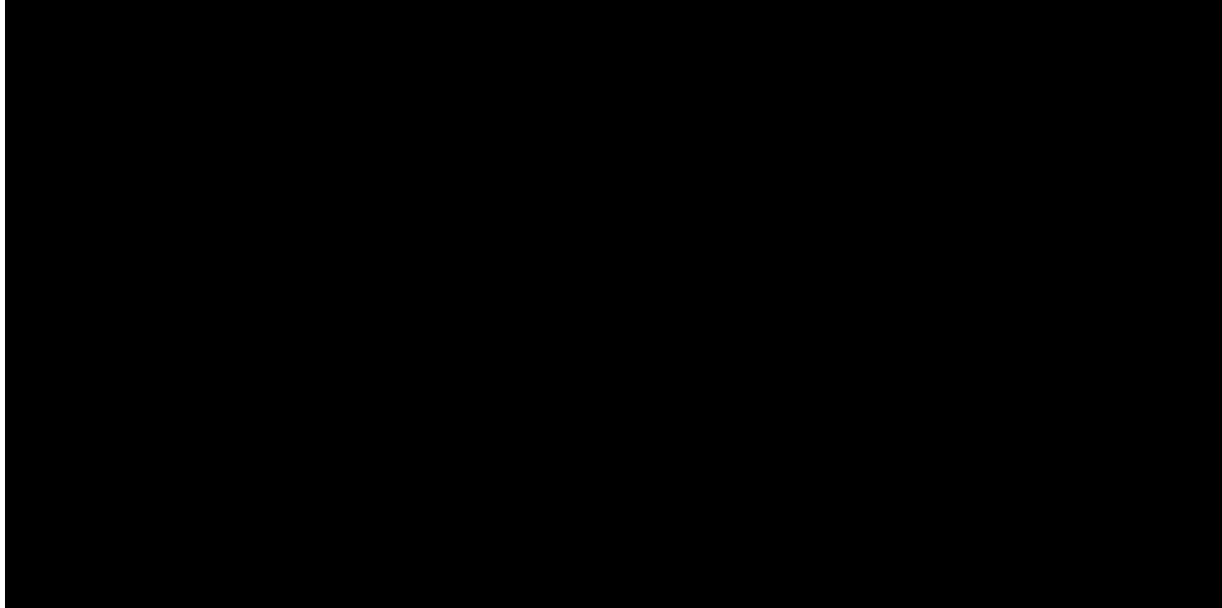
-  **テキストベース**
 - Markdownでコンテンツに集中し、後でスタイルを整えられる
-  **開発者フレンドリー**
 - コードハイライト、ライブコーディングと自動補完
-  **インタラクティブ**
 - コンポーネントを埋め込んで表現ができる
-  **録画**
 - 組み込みの録画とカメラビュー
-  **ポータブル**
 - PDF、PNG、またはGitHub PagesにSPAをエクスポート可能
-  **カスタマイサブル**
 - ウェブページで可能なことは何でもできる

Slidevについてもっと読む

⭐ Slidev



AI Workerの紹介（4月）



AI Workerの利用言語

TS

Frontend | Backend | Scripts

Languages



- **TypeScript** 89.7%
- **Jinja** 4.7%
- **EJS** 2.3%
- **JavaScript** 1.7%
- **HTML** 0.5%
- **Bicep** 0.5%
- **Other** 0.6%



開発環境

BEFORE

-  IntelliJ Idea
 - IDE完成度が高い
-  VSCode
 - Extensionが充実

AFTER

-  Cursor
 - AI機能はGitHub Copilotを大幅に超えています
 - ユーザー体験が非常に優れています
 - OpenAIからの投資を受けて急速に成長中
 - VSCodeの全ての拡張機能と設定が利用可能
 - フォークであり、定期的に更新が行われる
 - \$20/月のコストは使用開始20分で元が取れるほどのある価値がある

✓ タスク管理

BEFORE

-  JIRA
 - 適しているタスク管理
 - 使用を望む者は少ない
-  GitHub Issues
 - 開くのが面倒
 - 機能が不足している

AFTER

-  Linear
 - エンジニアに選ばれる理由
 - 優れたユーザーエクスペリエンス
 - Notionを超える操作性
 - 非常に高速なレスポンス
 - Local First Architectureを採用

ブラウザ

BEFORE

-  Chrome
-  Safari

AFTER

-  Arc
 -  GitHubのプルリクをリアルタイムで表示されるLive Folders機能がリリースされた
-

 JS バンドラー BEFORE

-  Webpack
 - 2014年に 日本語初の記事を書きました

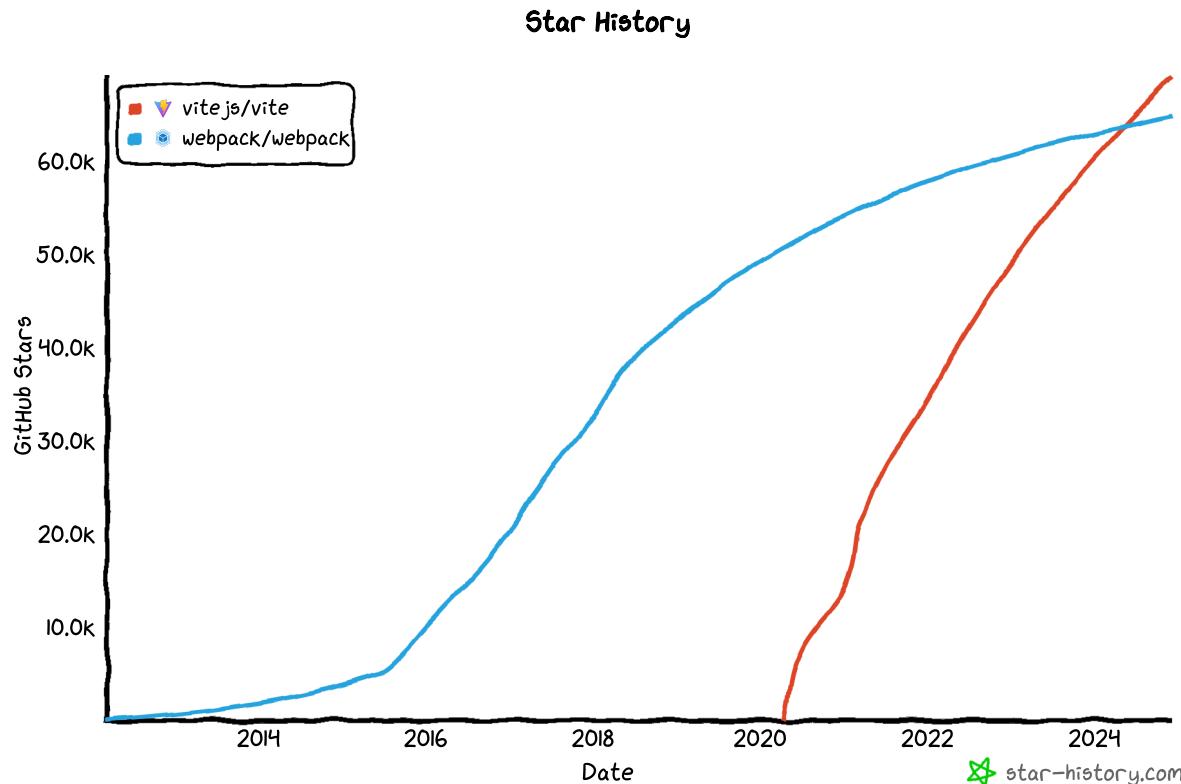
 AFTER

-  Vite
 - 一部Rust製
 - 開発時に速い

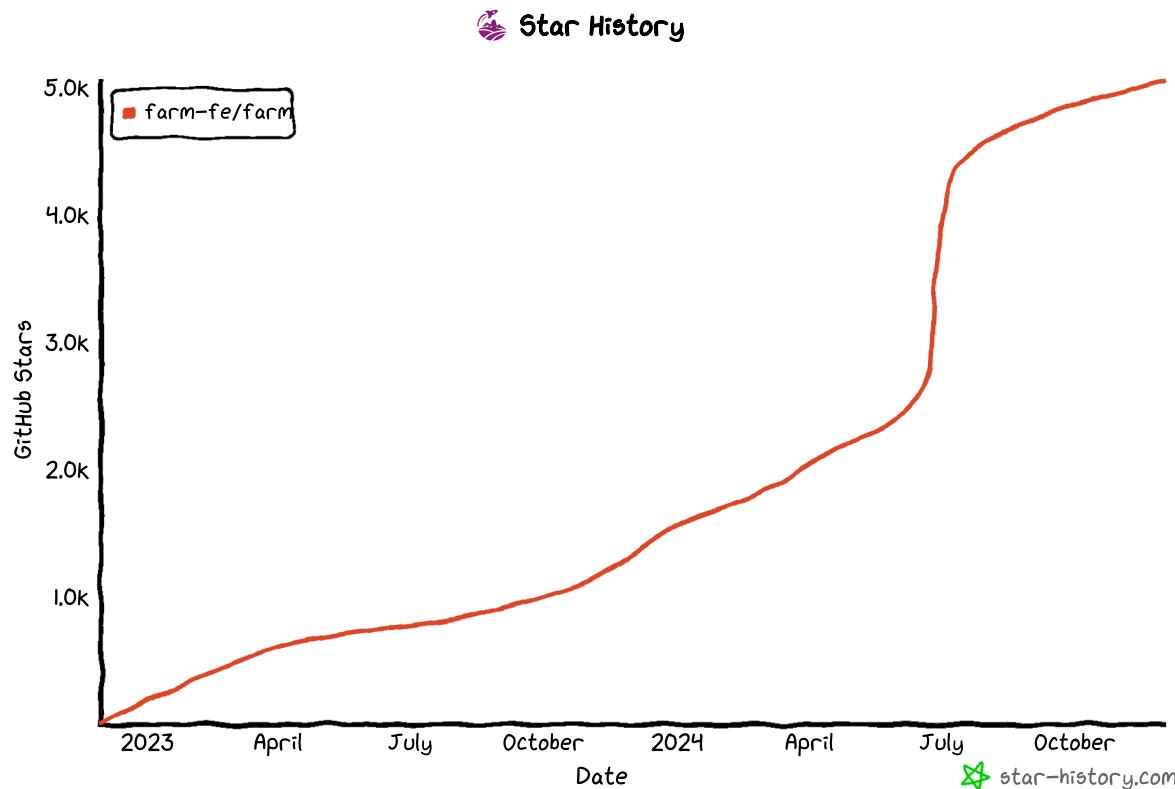
 FUTURE

-  Farm
 - Rust製
 - 100% Vite互換性
 - どんな時でも速い
 - v1.0がリリースされた

⭐ webpack → Vite



⭐ Farm



 CSS バンドラー

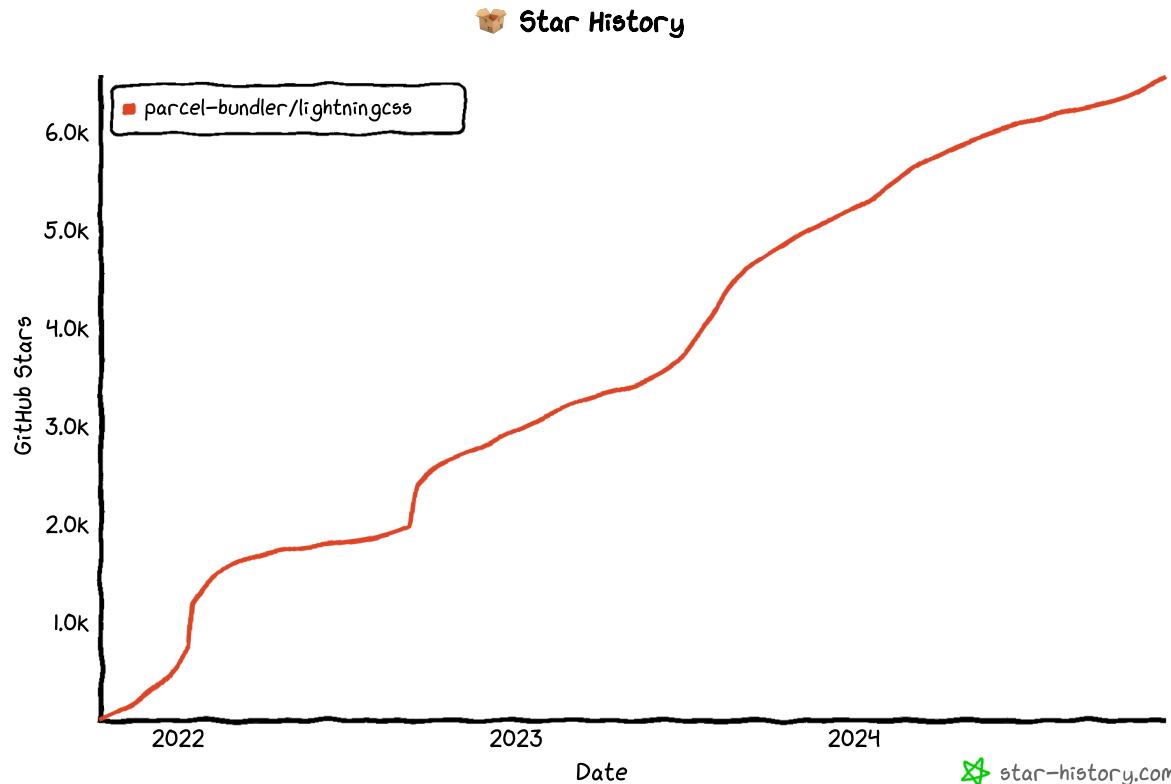
⌚ BEFORE

- » ESBuild

➡ AFTER

- ⚡ Lightning CSS
- Transpilation
- CSS Modules
- Bundling
- Minification
- Rust製、速い
- Viteでも使える

⭐ Lightning CSS





英語Linter

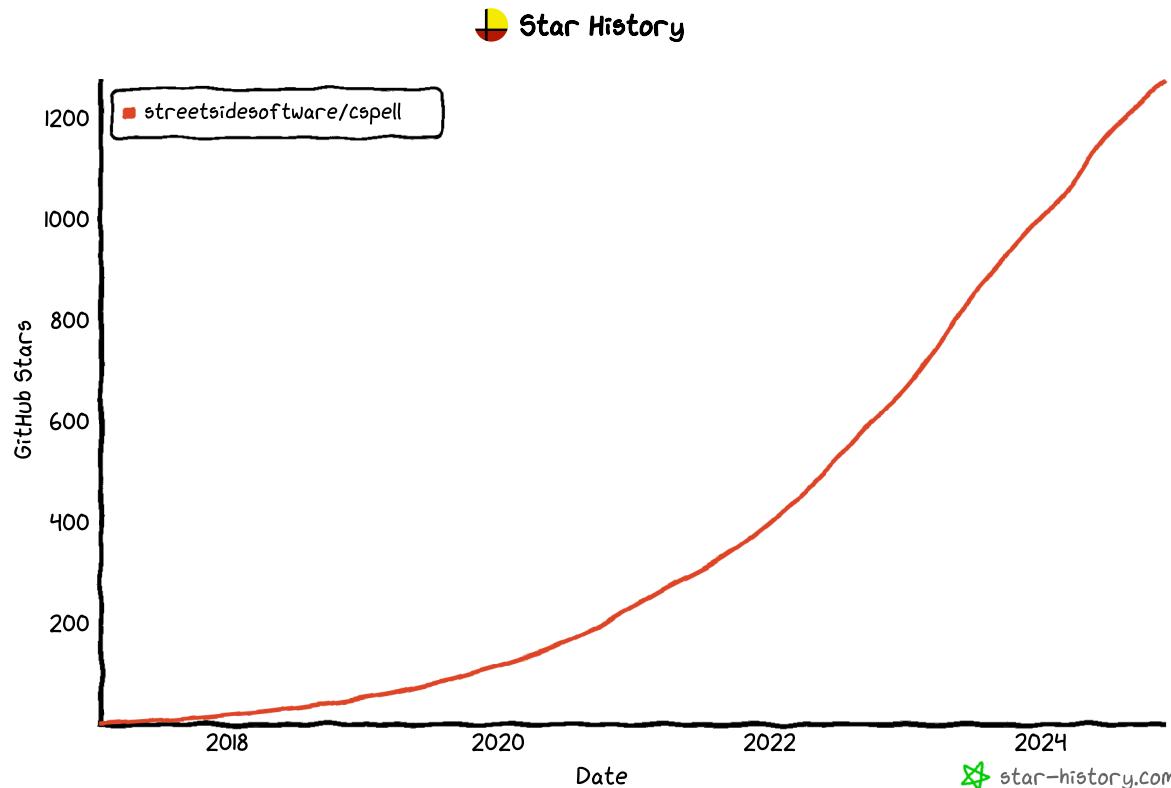
BEFORE

- 都度PR人間が修正
- 抜け漏れ発生

AFTER

-  CSpell
 - VSCode Extensionでリアルタイムチェック
 - Lefthookでコミット前にチェック
 - CI時でPRチェック

⭐ CSpell



☁ クラウド（最も大事）

 Cloudflareはもう「Edge CDN」ではない。
競争力ある立派なクラウドベンダーに進化。

⌚ BEFORE

-  GCP (40リージョン)
 -  DC : 東京、大阪
 -  DC : 無し
-  AWS (33リージョン)
 -  DC : 東京、大阪
 -  DC : 無し
-  Azure (60リージョン)
 -  DC : 埼玉、大阪
 -  DC : 3つ

➡ AFTER

-  Cloudflare (310リージョン)
 -  DC: 4つ (東京、大阪、福岡、那覇)
 - **Region Earth** :  DC: 37つ、 DC: 53つ
 -  コストメリットが顕著
 -  デプロイ速度が速い (10倍~)
 -  ストレージエグレス料金が不要
 -  I/O操作時の課金なし (LLM呼び出し等)
 -  Telemetryが無料
 -  GPUとLLM推論
 -  シンプル、楽しい！ (重要)



Cloudflareで利用可能になったもの

- **D1 Database (GA)**: SQLiteベースで使いやすいマルチテナント分散DB。
- **Hyperdrive**: 既存のDBを分散型化により体感速度を上げてくれる機能。
- **Worker Analytics (GA)**: ClickHouseベースの、Prometheusよりも速く、低コストな時系列DB。
- **Queues**: 無料のエグレスで、保証された配信でメッセージを送受信。
- **KV Bindings**: 環境変数内で動作。
- **AI (GA)**: Llama 3をサポートし、GPU上で推論を行い、Bring Your Own LORAsも可能に。
- **AI Gateway**: Claude、Azure、Bedrock、Vertexを追加サポート。
- **Workers Python**: FastAPI、Langchain、NumpyなどがCloudflare Workersで実行可能に。
- **Rate Limit**: APIのレート制限を簡素化するバインディング。
- **Cloudflare Media**: Zoomのようなビデオ会議ソリューションが自作可能に。
- **Tracing**: Baselineを買収し、無料のOpenTelemetryを提供。
- **Realtime**: PartyKitを買収し、Figmaのようなリアルタイムコラボレーションが簡単に開発可能。
- **RPC**: Cap'n Protoベースの、JSネイティブで簡単なゼロレイテンシーRPCシステムをWorkersで。



バックエンド言語

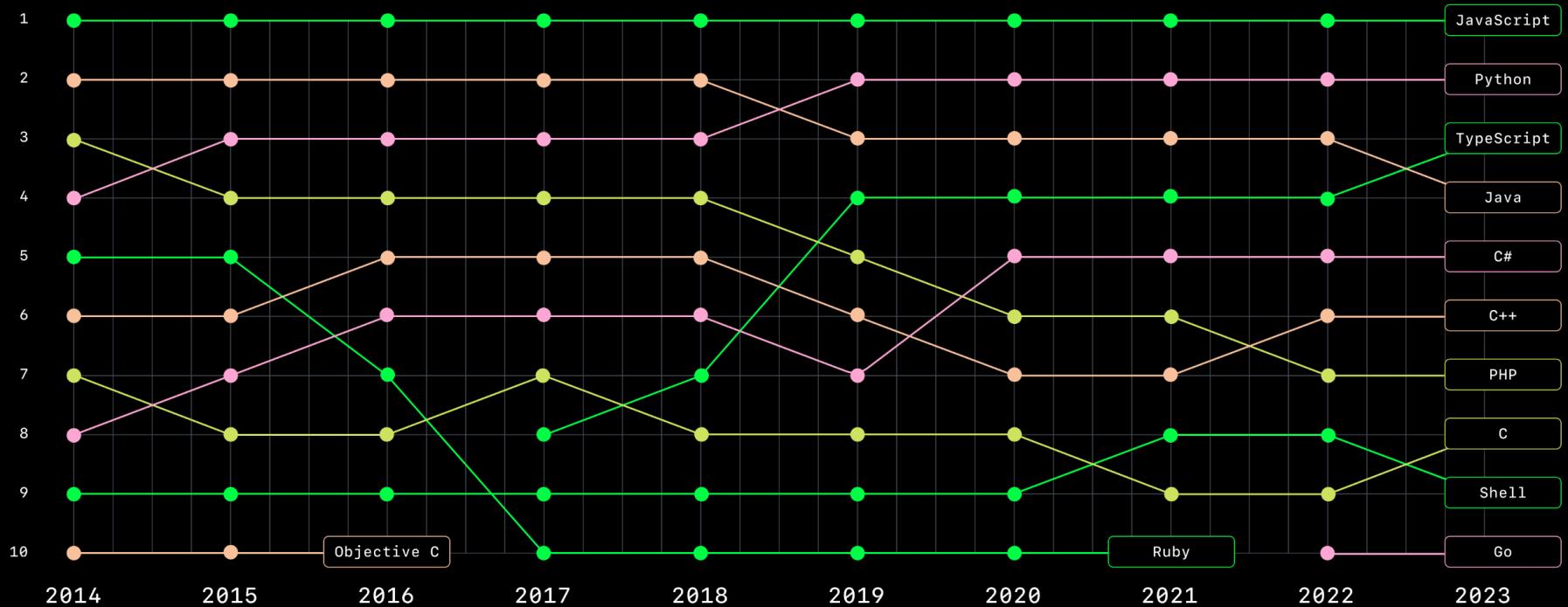
⌚ BEFORE

-  Golang

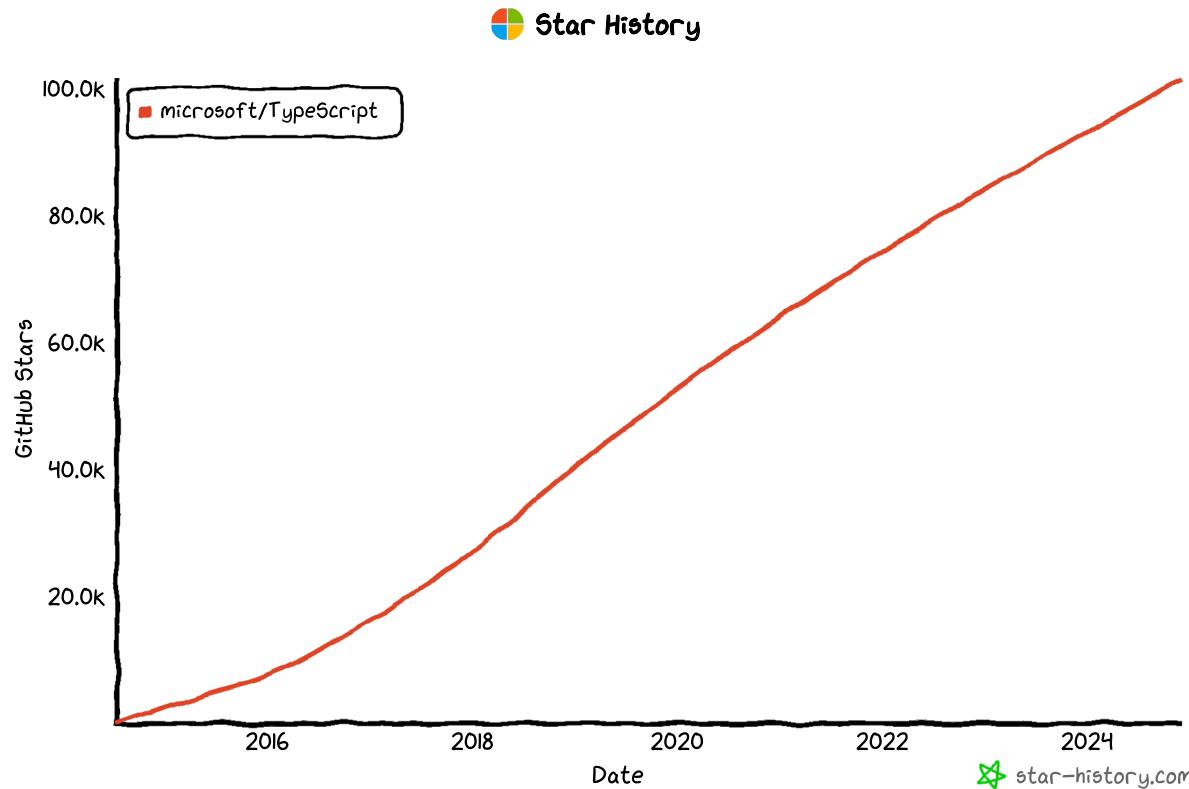
➡ AFTER

-  TypeScript

Top 10 programming languages on GitHub



⭐ TypeScript



特徴	TS		特徴	TS	
▪ 並行性と並列性	✗	✓	✖️ ブラウザで実行	✓	✗
⌚ ベーシックな型安全性	✓	✓	⬡ Edgeで実行	✓	✗
∅ Null安全性	✓	✗	iOS iOSで実行	▲	✗
❗ エラー安全性	✗	✓	Android Androidで実行	▲	✓
⤤ 配布のしやすさ	✓	✓	Pipes Pipes	✗	✗
⠡ Windows対応	✓	✓	ᴱ 代数的データ型	✓	✗
▶ スタンドアロンバイナリ	⠃⠃	✓	🔤 パターンマッチング	✗	✗



TypeScriptランタイム

⌚ BEFORE

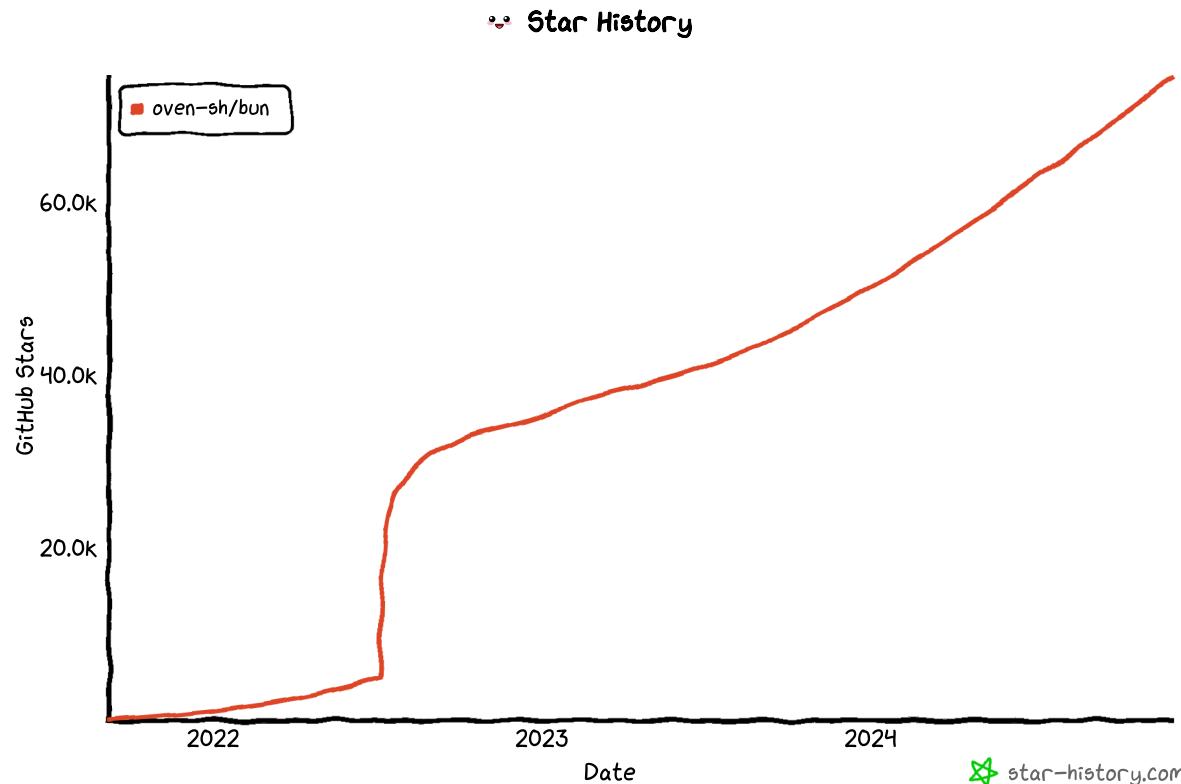
- Node



AFTER

- Bun
 - 実行時間が5倍速い
 - トランスペイブル不要
 - CJSとESMがミックス可能

⭐ Bun





TypeScriptパッケージマネジャー

⌚ BEFORE

- npm
- pnpm

➡ AFTER

- Bun
 - 10秒でインストール
 - npmより30倍速い

バージョンマネジャー

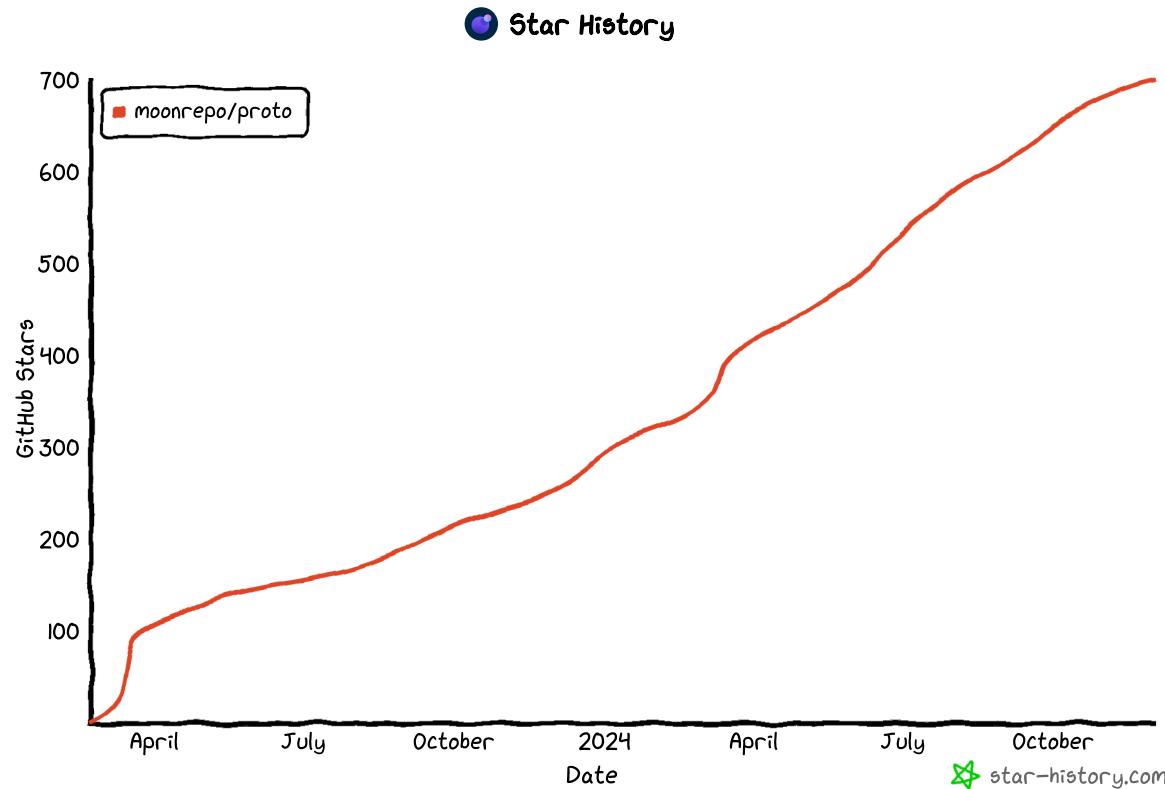
BEFORE

-  管理なし
-  Brew

AFTER

-  proto
 - Rust^製
 - Direnvで自動インストール
 - 対応ツール (50~)
 -  proto install bun
 -  proto install node
 -  proto install go
 -  proto install rust
 -  proto install python
 - ...

⭐ Proto





TypeScript テストランナー

⌚ BEFORE

- Vitest



AFTER

- Bun
 - 5倍速い

➤- スクリプト

⌚ BEFORE

-  Bash
-  Zx

➡ AFTER

-  Bun Shell
 - クロスプラットフォーム
 - ロジックが書きやすい
 - 若手も分かる
 - Full TS

```
import { $ } from "bun"

const output = await $`ls -l`.text()
console.log(output)
```

↑ モノレポCI

⌚ BEFORE

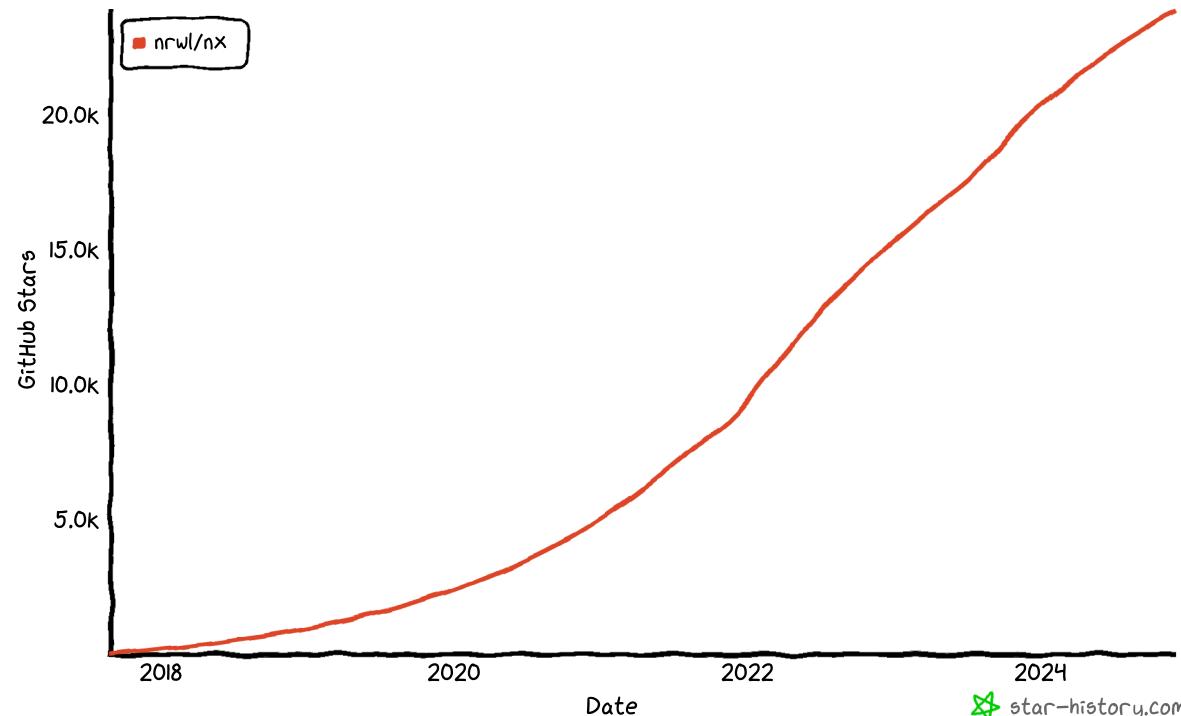
-  Turborepo
-  moonbase

➡ AFTER

-  Nx
 - 最も機能が充実している
 - 最もスケールする
 - 並行処理が得意
 - 速い



Nx Star History



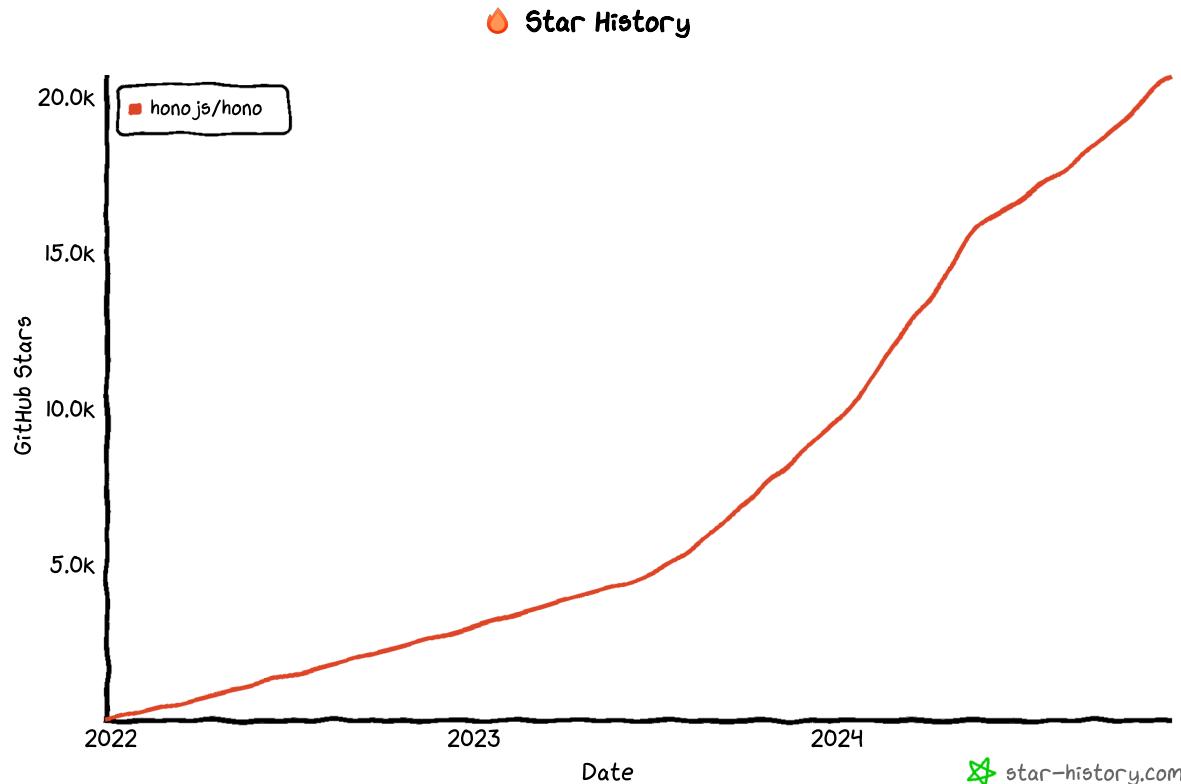
 サーバーフレームワーク BEFORE

- eX Express
-  Go

 AFTER

-  Hono
-  軽量、速い
-  シンプル
-  ミドルウェアが豊富
-  マルチランタイム
 -  ロックインされない
-  Bun
-  Node.js
-  Lambda
-  Vercel
-  Cloudflare Workers

⭐ Hono



Who is using Hono?

たくさん！

- cdnjs API Server
- Deno Docs
- Ultra
- Cloudflare
- Workers SDK
- Prisma
- Waku
- Unkey
- Drivly
- SticAI
- Skill Struck
- Reejs
- toddle
- LanderLab
- OpenStatus
- Loglib
- Use Scraper
- repeat.dev
- Nodecraft
- Hwy
- Unkey
- AI.LS
- ExpenSee
- Vocs
- Goens
- Trigger.dev
- NOT A HOTEL
- CyberAgent
- AI shift
- PartyKit
- Azule.
- OnlineOrNot
- Others!



データベースORM

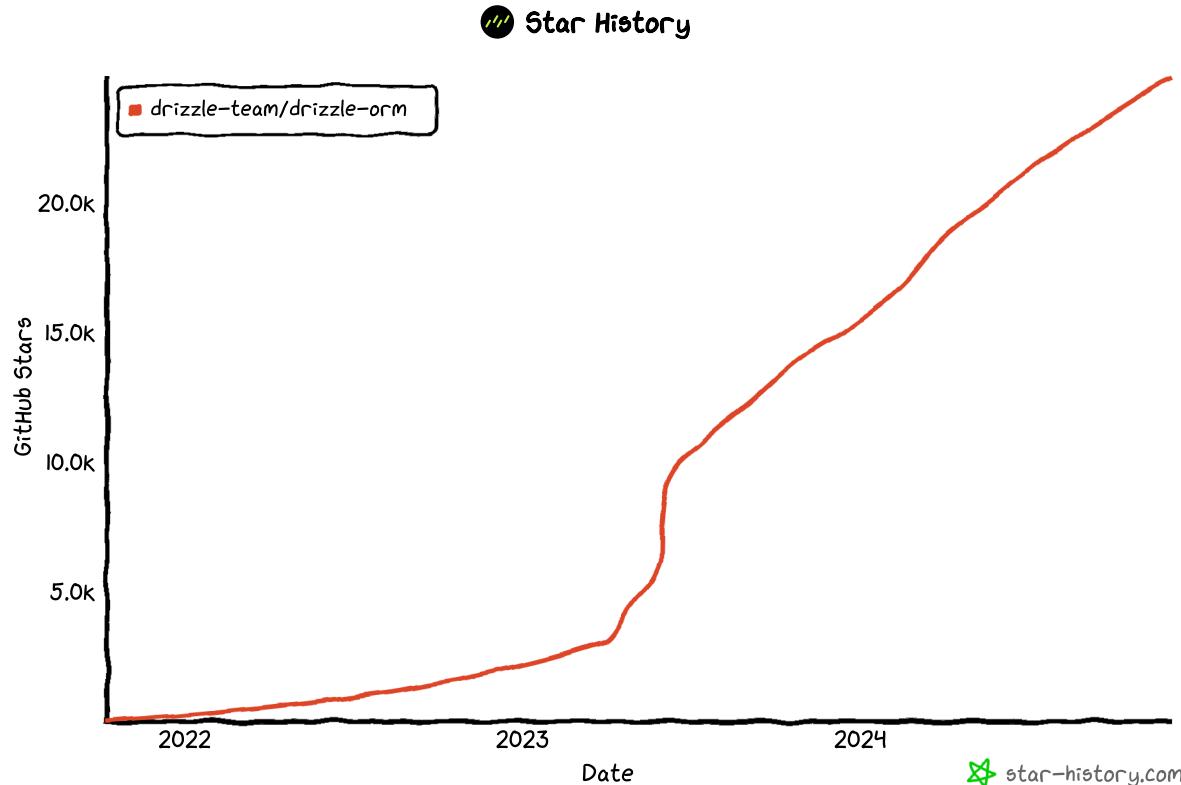
BEFORE

- Prisma

AFTER

- Drizzle
 - 軽量、速い
 - SQLに近い
 - マイグレーションできる
 - マルチランタイム
 - Bun
 - Vercel
 - Cloudflare Workers
 - Expo
 - Browser
 - Supabase
 - Electron
 - React Native
- マルチデータベース
 - PostgreSQL
 - Supabase
 - Vercel
 - SQLite
 - MySQL
 - Xata
 - Turso
 - Neon

⭐ Drizzle





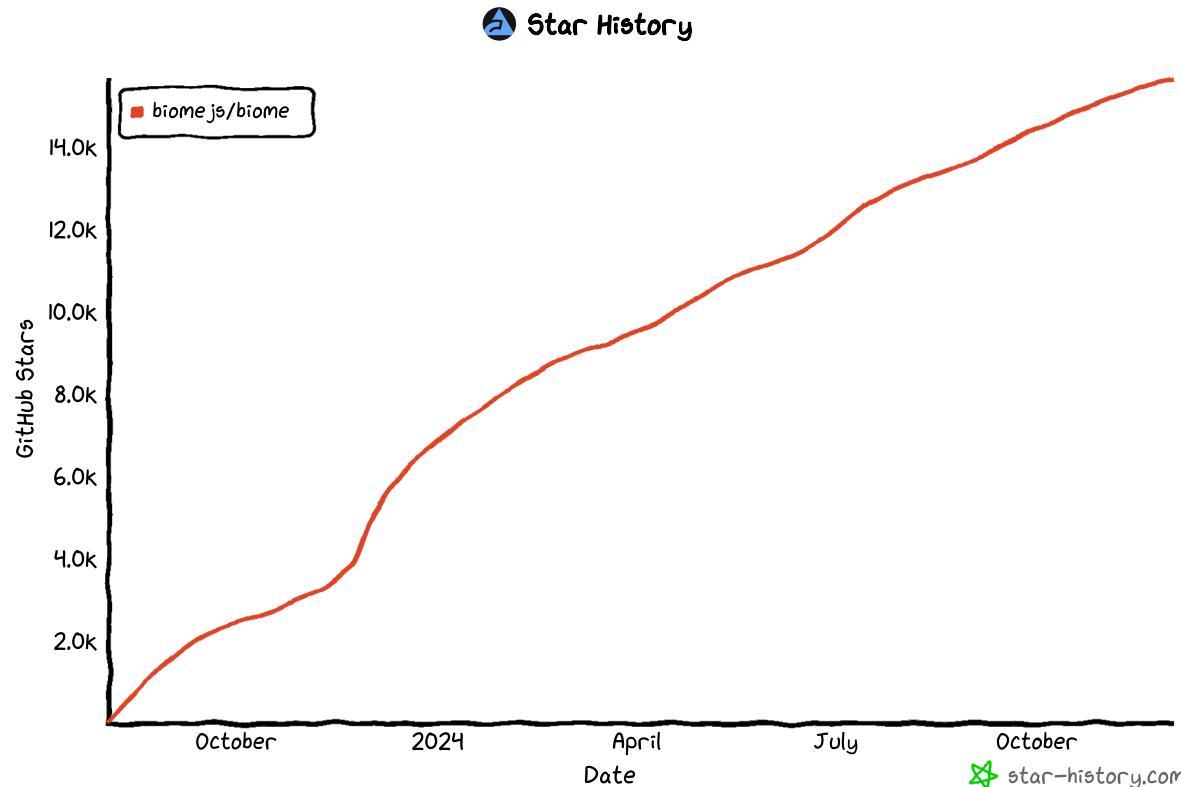
⌚ BEFORE

-  Eslint

➡ AFTER

-  Biome
 - Rust製
 - ESLint等200以上のルール
 - 速い

⭐ Biome





Formatter

⌚ BEFORE

-  Prettier
-  dprint

➡ AFTER

-  Biome
- Rust製
- Prettierと97%互換性
- 35倍速い

🚩 Feature Flags

⌚ BEFORE

- 🛡️ 自作
- ➔ LaunchDarkly
- 🔐 ConfigCat
- 🔥 Firebase Remote Config
- 📈 Bugsnag

➡ AFTER

-  DevCycle
 - 50ms以下のレイテンシ
 - SDKの豊富さ: 導入が容易
 - 料金体型: MAU課金、価格面で良心的な料金
 - 使いやすさ: DX・UXが直感的
 - リアルタイム更新: SSE経由
 - OpenFeature対応: ロックインを防げる
 - IDEのExtension: VSCodeのExtension
 - Edge Flags: Edge DB機能の提供
 - Local Bucketing



認証基盤

BEFORE

- 🔥 Firebase Auth
- 🔖 Supabase Auth
- ⭐ Auth0
- 🌐 Okta

AFTER

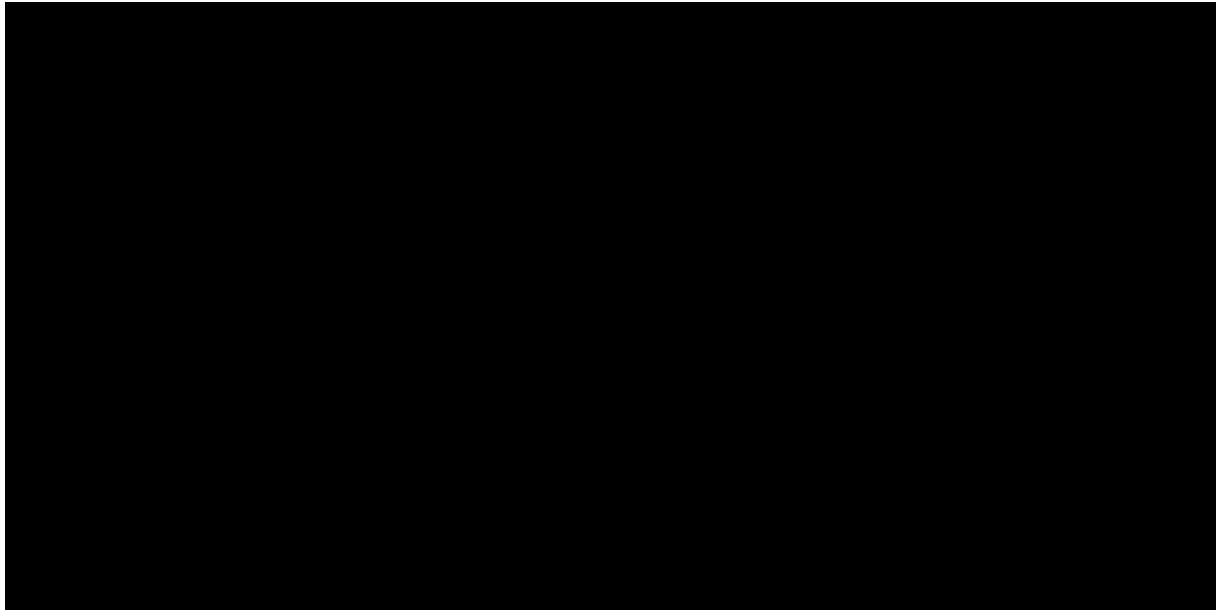
- Clerk
- ✨ 最も優れたユーザーエクスペリエンス
- 💰 合理的な価格設定
- 🚀 生産性向上
- 🎯 React コンポーネントを提供
- 🚑 リードタイムの短縮
- stripe 今後、Stripeとの連携予定

UXよりも低コストが重要な場合

- Kinde
- Lucia

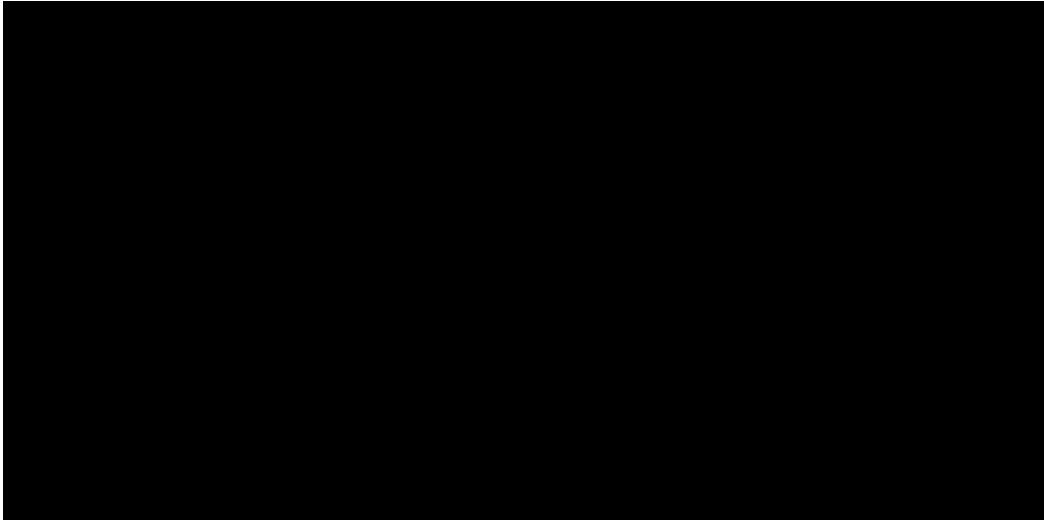


自作したCLI自動LLM翻訳





Cloudflare Workers上で動作するLLM



直近の動向

堅牢なTypeScriptを

▪ Effect

- Golang/Rustも羨ましくない
- 欲しかったスタンダードライブラリ
- Reactでポピュラーになった関数型
- 完璧なエラーハンドリング
- リトライ処理
- 並行処理
- パターンマッチ
- スキーマ
- シリアライゼーション
- トレーシング



Effect



The best way to
ship faster
in TypeScript

Introduction to Effect

▶ Watch Video

Next Generation TypeScript

後半: マジック編

魔法のような体験が、
製品の価値を高める。

A background image featuring a cartoon character with large blue eyes, a small tuft of hair, and a white and orange polka-dot bow. The character has a wide, joyful expression with a wide-open mouth showing pink tongue and teeth. A large, glowing yellow speech bubble originates from the character's mouth, containing the text "iOS 18のアニメーション". The background is a soft-focus gradient of light colors with scattered golden sparkles.

iOS 18のアニメーション

✨ 魔法のような体験が、
製品の価値を決定的に高める

成功事例 🏆

- Apple iOS 18 🍎 の流麗なアニメーション
 - 🔍 ユーザーを魅了する完璧な動き
 - ⚜️ プレミアム体験の象徴に
- Pixar & Disney 🎬 の哲学
 - 💫 細部へのこだわりが感動を生む
 - 💎 最高の品質が最高の価値を生む

Webの可能性 🚀

- 🎮 ゲーム業界では当たり前の表現力
- 📱 ネイティブアプリの洗練された動き
- しかし、Webではまだ未開拓 🌐
 - ⚡ 技術的には十分可能
 - 💫 差別化の大きなチャンス

こんな経験ありませんか？✨

- 😞 素敵なアニメーションを実装したいけど…
- 😵 WebGLやシェーダーは難しそう…
- 🕒 学習に時間がかかりすぎる…
- 💭 ゲーム開発じゃないから諦めるべき？
- 📚 数学の知識が足りない…



フロントエンド表現の進化 🚀

2010年代前半

 CSS Animations

 **jQuery** jQuery効果

 Canvas 2D

2010年代後半

 CSS 3D Transforms

 WebGL 登場

 Three.js台頭

2020年代

 WebGL2.0

 WebGPU

 シェーダー表現

従来の開発フロー



1. 基礎学習 (1-2週間)

- WebGL基礎
- GLSL文法
- 数学の復習

2. 環境構築 (2-3日)

- ボイラープレート
- シェーダーローダー
- デバッグツール

3. 実装 (1-2週間)

- 試行錯誤
- パフォーマンス調整
- クロスブラウザ対応

魔法使いの開発フロー



1. アイデアの具体化 (30分)

- エフェクトの明確化
- 参考実装の収集
- 要件の整理

2. Cursorとの対話 (30分)

- コード生成
- デバッグサポート
- 最適化アドバイス

3. 実装＆調整 (1時間)

- 即座の試行錯誤
- リアルタイム改善
- パフォーマンス最適化

実例：ゴールデンタイトル with Sparkles ✨

従来の開発

- 学習期間：3-5日
- 実装期間：5-7日
- デバッグ：3-5日
- 合計：**11-17日** 😱

Cursor活用

- 要件定義：30分
- 実装：1時間
- 調整：30分
- 合計：**2時間** 🚀

具体的な開発プロセス

```
uniform float time;
uniform vec2 resolution;
varying vec2 vUv;

void main() {
    vec2 uv = gl_FragCoord.xy / resolution;
    // ここでエフェクトの計算
}
```

Cursorの活用ポイント

1. コードの生成
2. エラーの解決
3. 最適化提案

重要な考慮点

1. パフォーマンス
2. ブラウザ対応
3. フォールバック

魔法を使う場面



ECサイト

- 商品展示の演出 - インタラクティブなUI -
購買意欲の向上



メディア

- 没入感のある記事 - データビジュアライ
ゼーション - スクロールアニメーション



企業サイト

- ブランド体験 - プロダクト紹介 - ランディ
ングページ

レンダリング選択の戦略



SVG/CSS

- 単純な形状
- 文字アニメーション
- 軽量なエフェクト

Canvas 2D

- 中程度の複雑さ
- パーティクル（少量）
- 基本的な描画

WebGL

- 複雑なエフェクト
- 大量パーティクル
- シェーダーエフェクト

選択基準

1. パフォーマンス要件
2. 表現の複雑さ
3. 対象デバイス
4. 開発期間

パフォーマンスの魔法 ⚡

最適化のポイント

1. GPUの活用
 - シェーダーの最適化
 - バッチ処理の活用
2. メモリ管理
 - テクスチャの圧縮
 - オブジェクトプーリング
3. レンダリング制御
 - ビューポート外の停止
 - 解像度の動的調整

モニタリング指標

1. FPS (Frames Per Second)
2. CPU/GPU使用率
3. メモリ消費
4. 電力効率

```
// パフォーマンスマニタリング例
stats.begin();
render();
stats.end();
```

フォールバックの重要性



想定シナリオ

- WebGL非対応
- 低スペックデバイス
- 省電力モード
- 古いブラウザ

対応戦略

1. 段階的な機能低下
2. 代替表現の用意
3. ユーザー通知
4. パフォーマンス測定

魔法使いへの道のり



1. **基礎を理解** レンダリングの仕組み、GPU活用の原則
2. **Cursorと対話** 実装のサポート、最適化のアドバイス
3. **実験と検証** 様々なアプローチを試す
4. **最適化** パフォーマンスとUXのバランス
5. **フィードバック** ユーザー反応の測定と改

ベストプラクティス



技術選択

- 目的に応じた手法
- スケーラビリティ
- メンテナンス性
- チーム共有

注意点

- バンドルサイズへの影響
- 初期読み込み時間
- アクセシビリティ
- SEO対応

プロジェクト管理

- 段階的な導入
- A/Bテスト
- パフォーマンス計測
- ドキュメント化

未来への展望



WebGPU時代
より高度な表現が可能に



AI支援の進化
開発効率の更なる向上



クロスプラットフォーム
あらゆる環境で魅せる

Call To Action 🚀

今日から、あなたも魔法使いになれる



まずは小さく
簡単なエフェクトから始めよう



恐れずに挑戦
AIが常にサポート



表現の幅を広げよう
ユーザーを魅了する体験を

実験の場：友人の依頼 



イベント招待状

要件定義

- 手書き風の名前表示
- 魔法のような演出効果 
- イベントLP制作 
- 決済システム連携 

実験的な挑戦

- シェーダーによる光の演出
- 手書きフォントのアニメーション
- インタラクティブな演出
- スムーズな決済フロー

重要な気づき

本業では試せない実験が、個人開発で可能に

- ✓ 新技術の検証
- ✓ 失敗を恐れない挑戦
- ✓ 知見の蓄積

① 個人開発での成功体験を、実務のサービスに還元できる

実験 → 実践のサイクル



個人開発での実験

- リスクを恐れない - 新技術への挑戦
- 自由な発想



知見の獲得

- 実装ノウハウ - パフォーマンス対策 -
- 運用の注意点



実務への還元

- 検証済みの技術 - 具体的な改善提案
- 説得力のある提案

実例



実装事例 { }

デザイン 🎨

- レイアウト&フォント: Canva
 - Figmaより迅速な開発
 - 豊富なテンプレート
 - デザイン初心者でも使いやすい

画像生成 🖼

- 背景写真: MidJourney 
 - 高品質な画像生成
- リアル系画像: AI SCREAM Flux Pro
- プロンプト生成: Claude 3.5 

フロントエンド

- WebGL: Three.js 
- シェーダー実装: Cursor Composer
- SSG: Astro  **astro**
- Client: React 
- 配信: Cloudflare Pages  **CLOUDFLARE**

バックエンド

- LINEボット: Astro Functions + Workers
- メール送信: Workers + Resend API
- 決済: Stripe Button **stripe**

AIアシスタントの使い分け



Claude 3.5 AI

- デザイン感覚が優れている
- 数値計算は不得意
- 時々実装を壊すことも
- クリエイティブな提案力

OpenAI o1-preview

- 理屈的な思考
- エラーの少なさ
- 大規模設計が得意
- 正確性重視

💡 Tip: 両方のアシスタントを試行錯誤しながら使うのがベスト

0→1フェーズの新しいカタチ💡

アンチパターン ✗

- ❗ 従来の分業制
- ⌚ コミュニケーションコスト
- ⇄ 意思決定の遅延

生成AI時代のベストプラクティス ✓

- 🚀 フルスタックスキル
- 🎨 デザインセンスの習得
- ⚡ 迅速な実装と改善

★ フロントエンドエンジニアへの警告

- ✓ デザインスキルを磨く
- ✓ AIツールを使いこなす
- ✓ 0→1の全工程を理解する

⚠ もしフロントエンドエンジニアがAIツールを活用しなければ、他の職種の方がフロントエンド開発を担うことになります。
「魔法使い」になるか、「魔法使い」に仕事を奪われるか、選択は我々次第です。

Thank You! ✨

Contact & Social

 @gunta85

 @gunta

 guntherbrunner.art