爆速でプロダクトをリリースしようと思ったらマイクロフロントエンドを選んでいた

フロントエンドカンファレンス東京 2025

@nkgrnkgr

自己紹介

Nokogiri (@nkgrnkgr)

- 株式会社カケハシ
- 生成AI研究開発チーム
- ソフトウェアエンジニア

薬局向け業務システム「Musubi」に生成AI機能を組み込む仕事をしています

Musubiの成功ストーリー

2021年：薬局業務のDX化

- 薬剤師が手書きしていた薬歴をデジタル化
- 業務時間を大幅に削減
- 全国の薬局に導入され市場シェアを獲得

カケハシの創業プロダクトとして大成功

しかし2024年、状況は一変

競合の動き

- A社：音声入力で薬歴を自動作成
- B社：AIが服薬指導を提案
- C社：過去の薬歴から次回指導を予測

Musubiだけが生成AI機能を持っていない

「今動かなければ市場を失う」

経営陣の決断

- 生成AI機能の開発を最優先事項に
- 専門チームを新設
- 3ヶ月以内のリリースを目標

スピードが全てを決める状況

新チームの結成

必要なスキルセット

- 音声認識技術
- 自然言語処理
- LLMの活用経験
- 医療ドメイン知識

既存のMusubiチームにはないスキル

集まったメンバーの特徴

PharmacyAIチーム（5名）

- 4名が入社1年以内の新メンバー
- 全員がReact/TypeScriptのエキスパート
- Angularの経験はほぼゼロ
- 生成AIを使った開発に慣れている

Musubiチーム

- 創業時からのメンバーが中心
- Angular/RxJSのエキスパート
- 医療システムの要件を熟知

組織構造がアーキテクチャを決めた

```
Musubiチーム                    PharmacyAIチーム
  ├─ Angular専門                  ├─ React/生成AI専門
  ├─ 創業メンバー                 ├─ 新規採用メンバー
  ├─ 安定運用重視                 ├─ 高速開発重視
  └─ 月次リリース                 └─ 週次リリース
```

別チーム・別リリースサイクル・別技術スタック

理想と現実のギャップ

理想

「PharmacyAIチームもAngularで開発して統合」

現実

- 新メンバーにAngularを学ぶ時間はない
- Reactで開発した方が3倍速い
- 生成AIのサンプルコードもReactが豊富

「きれいな統合」より「速いリリース」を選択

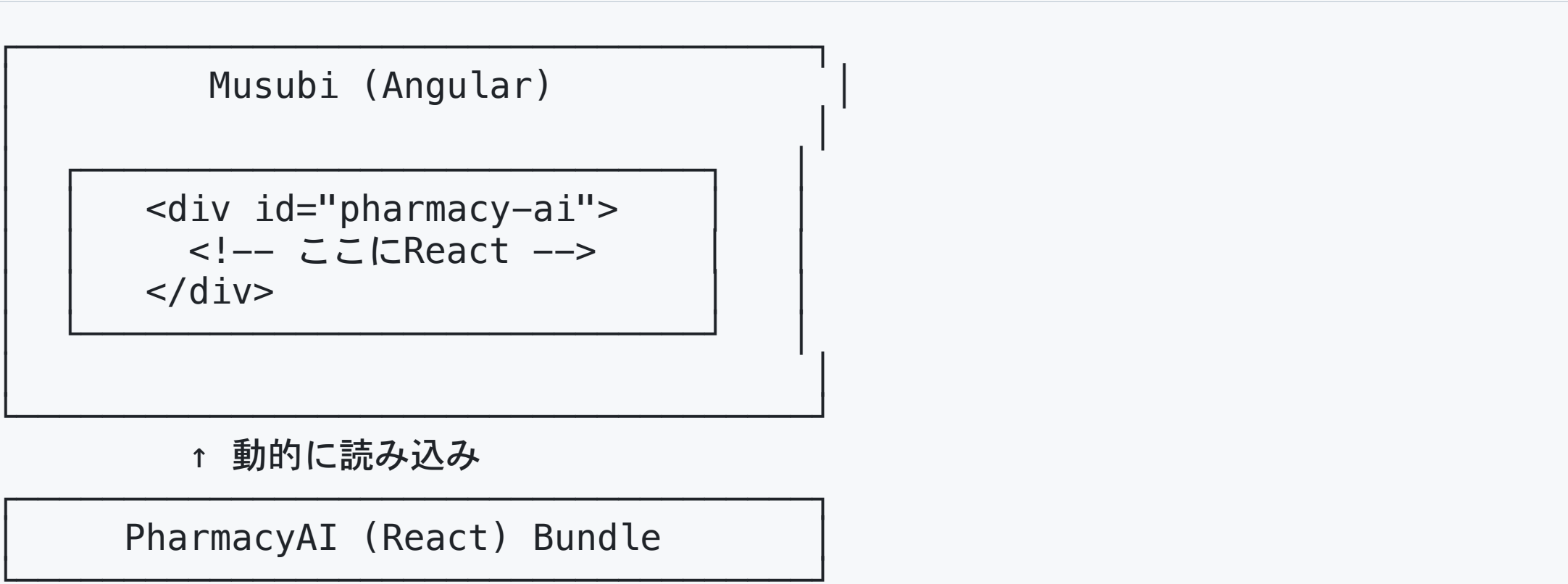
技術的な挑戦：2つのSPAをどう繋ぐか

検討した選択肢

1. **iframe** → セキュリティとUXの問題
2. **Web Components** → Angularとの相性問題
3. **Module Federation** → 3ヶ月では無理
4. **動的読み込み + CustomEvent** ← これを選択

一番シンプルで確実な方法

アーキテクチャ全体像



ReactアプリをAngularが呼ぶ仕組み

```
// Angular側のコード
export class MuSubiComponent implements OnInit {
  async loadPharmacyAI() {
    // ビルド済みのReactアプリを動的に読み込み
    const script = document.createElement('script');
    script.src = 'https://cdn.example.com/pharmacy-ai.js';

    const link = document.createElement('link');
    link.href = 'https://cdn.example.com/pharmacy-ai.css';
    link.rel = 'stylesheet';

    document.head.appendChild(link);
    document.body.appendChild(script);
  }
}
```

実際のCustomEvent活用例1：コンテキスト同期

```
// Angular側：ログインユーザーが切り替わった時
class UserService {
  switchUser(newUser: User) {
    // Angularの処理...

    // PharmacyAIに通知
    window.dispatchEvent(new CustomEvent('musubi:context-changed', {
      detail: {
        type: 'user-switch',
        user: { id: newUser.id, name: newUser.name }
      }
    }));
  }
}
```

非同期で動く2つのアプリの状態を同期

実際のCustomEvent活用例2：患者情報の同期

```
// Angular側：表示中の患者が変わった時
class PatientViewComponent {
  @Input() set patient(value: Patient) {
    this._patient = value;

    // PharmacyAIの表示も更新する必要がある
    window.dispatchEvent(new CustomEvent('musubi:patient-changed', {
      detail: {
        patientId: value.id,
        patientInfo: this.sanitizeForExternal(value)
      }
    }));
  }
}
```

実際のCustomEvent活用例3：データ転記の泥臭い実装

```
// React側：生成した薬歴をMusubiに転記
const transferToMusubi = async (prescription: string) => {
  let retryCount = 0;
  const maxRetries = 5;

  const tryTransfer = () => {
    window.dispatchEvent(new CustomEvent('pharmacy-ai:transfer-data', {
      detail: { prescription, requestId: generateId() }
    }));
  };

  // Angularのコンポーネントが準備できていない可能性
  const checkResponse = setTimeout(() => {
    if (retryCount++ < maxRetries) tryTransfer();
  }, 1000);
};
```

型安全性とデバッグの工夫

型定義の共有

```
// shared-events パッケージで型を共有
type EventMap = {
  'musubi:context-changed': ContextChangeEvent;
  'pharmacy-ai:transfer-data': TransferDataEvent;
};
```

デバッグツール

```
// 開発環境でイベントを可視化
if (isDevelopment) {
  console.group(`🔔 ${event.type}`);
  console.log('Detail:', event.detail);
  console.groupEnd();
}
```

デモ：実際の動作を見てみましょう

シナリオ

1. 薬剤師が患者を選択（Musubi）
2. 音声録音を開始（PharmacyAI）
3. AIが音声を文字起こし
4. 薬歴を自動生成
5. Musubiに転記して保存

ユーザーから見ると1つのアプリケーション

成果と教訓

成果

- **3ヶ月でリリース**（通常の半分）
- **各チームが最大パフォーマンス**を発揮
- **市場での競争力を回復**

教訓

- **CustomEventはシンプルで強力**
- **型安全性の確保**が重要
- **デバッグの仕組み**を最初から

ビジネス → 組織 → 技術 の順で選ぶ

ご清聴ありがとうございました

質問・議論したいこと

- CustomEventでのフレームワーク間連携
- マイクロフロントエンドの運用Tips
- 組織構造と技術選択の関係

@nkgrnkgr