sansan

TypeScriptを活用した型安全なチーム開発 2024

React Routerで実現する型安全なSPAルーティング

技術本部 Eight Engineering Unit

鳥山らいか @pvcresin

鳥山 らいか / @pvcresin

Sansan株式会社
技術本部　Eight Engineering Unit

2019年 Sansan株式会社に新卒入社。

Eightのフロントエンドの開発支援や技術的改善を行う。

その一環として、フロントエンドのフルTS化を主導。

直近では、開発者の生産性向上に向き合うDPEチームで
Rubyに型をつけている。

アジェンダ

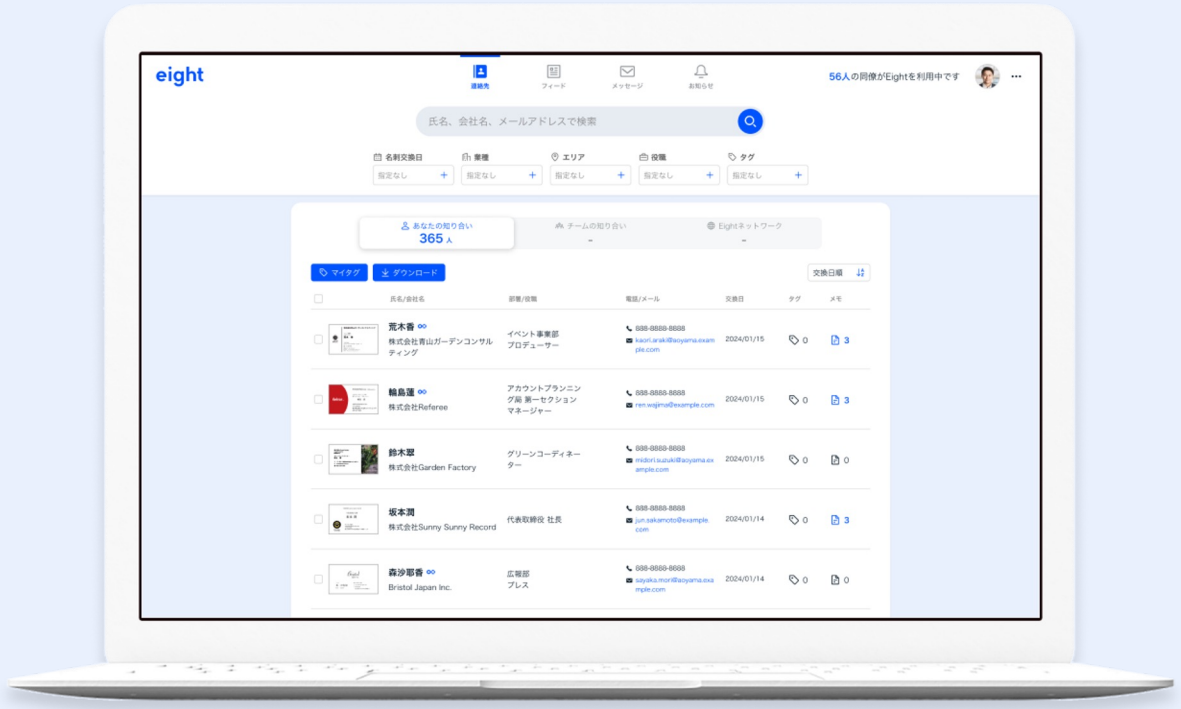
- Eightの紹介
- 課題①: 不正なURLへの遷移
- 課題②: ルーティング変更時の修正の多さ
- まとめ

タッチで交換。スマートに管理。

名刺アプリ「Eight」

Web版 Eight

eight



企業向けサービス

eight team

個人の名刺をチーム単位で共有して管理できるサービス

eight career design

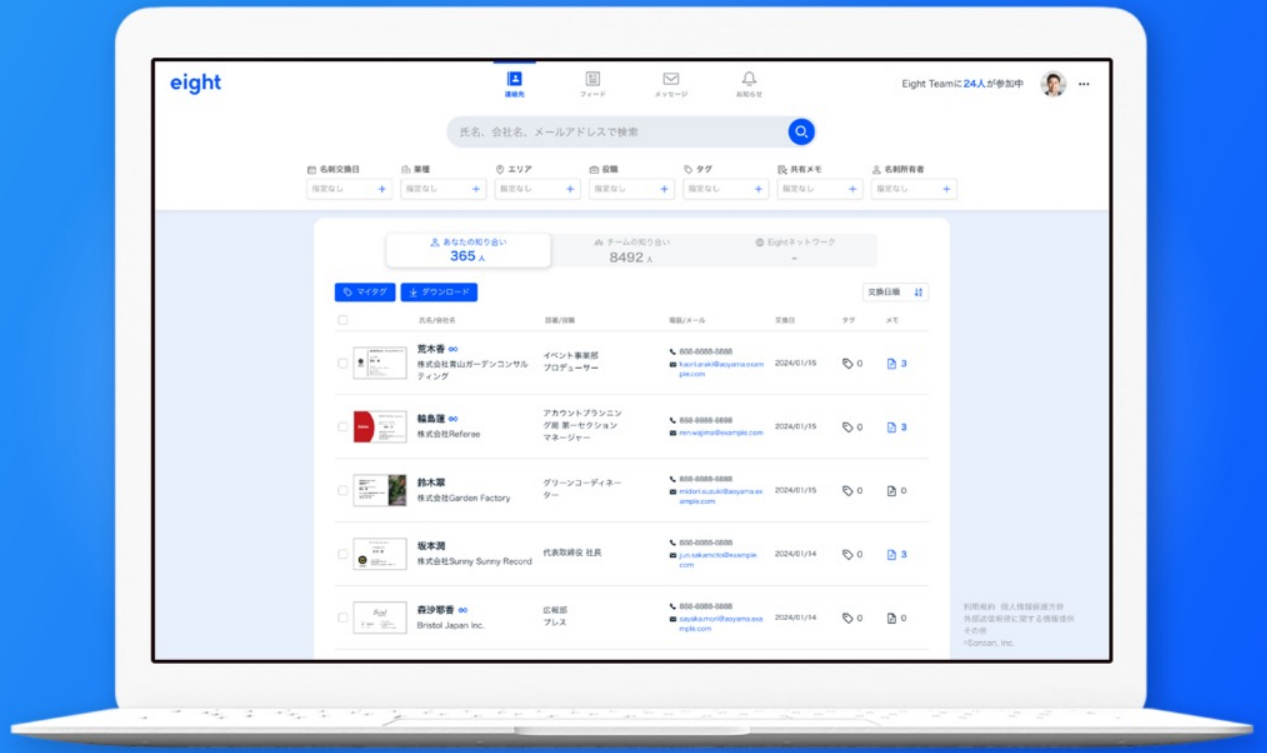
Eightユーザーにスカウトが送れる採用サービス

技術スタック

フロントエンド	TypeScript / React
バックエンド	Ruby / Ruby on Rails
インフラ	AWS
CI / CD	Circle CI / GitHub Actions

eight

名刺アプリ「Eight」Web版を大幅リニューアル



いろいろ大変な作業があった

- レイアウトの修正
- ページの新規作成
- ページのURL変更
- ページの削除

これらはすべてルーティングに絡む

➜ 型安全性について今一度整理しよう

React Router

React Router

- 宣言的にルーティングをかける君
- ネストやレイアウト、データ取得など +α な部分も引き受けてくれる

```
createRoot(root).render(
  <BrowserRouter>
    <Routes>
      <Route path="/" element={<Home />} />
      <Route path="/users/:id" element={<User />} />
    </Routes>
  </BrowserRouter>
);
```

課題①

不正なURLへの遷移

不正なURLへの遷移

- 文字列を直接書く形式だと、不正なURLの指定に気付きづらい

```
// typo
<Link to="/userz">here</Link>

// id is undefined
<Link to={`/users/${id}`}>here</Link>
```

- 個別に対応すると、URL生成のロジックが点在してしまう

URL生成ロジックをまとめる

URL生成ロジックをまとめる

- ルーティングの情報をまとめ、URL生成ロジックを集約

```tsx
// routes.tsx
export const ROOT = {
  HOME: "/",
  USER_ID: "/users/:id",
} as const;

export const homePath = () => ROOT.HOME;

export const userPath = ({ userId }: { userId: string }) =>
  ROOT.USER_ID.replace(":id", userId);
```

- とはいえ、もうちょっと楽に実現したい...

generatePath関数の導入

generatePath

- URLのパターンとパラメータを元に、URLを生成してくれる関数
- パラメータが足りない場合に型エラーを出す

```
generatePath("/users/:id", { id: "pvcresin" }); // "/users/pvcresin"
generatePath("/users/:id", { userId: "pvcresin" });

generatePath("/files/:type/*", {
  type: "img",
  "*": "cat.jpg",
}); // "/files/img/cat.jpg"
```

generatePathを組み込む

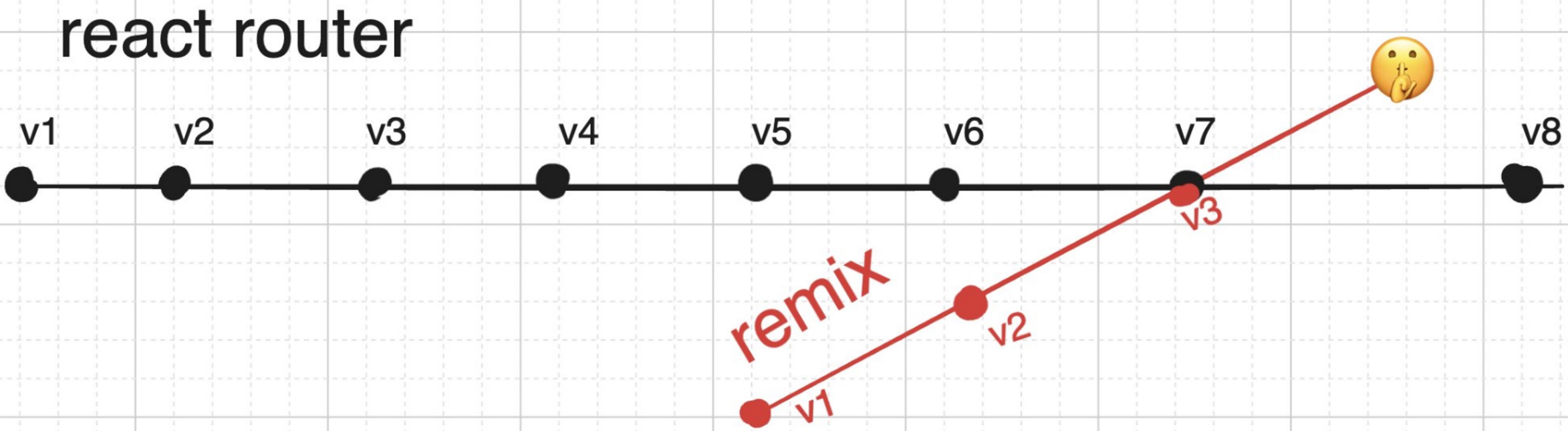
```ts
// before
export const userPath = ({ userId }: { userId: string }) =>
  ROOT.USER_ID.replace(":id", userId);

// after
export const userPath = ({ userId }: { userId: string }) =>
  generatePath(ROOT.USER_ID, { id: userId });
```

- 少しだが、型安全な部分が増えた
- URL内に複数のパラメータが存在する場合には、より恩恵が得られそう

課題②

ルーティング変更時の修正の多さ



出典: https://x.com/jacobmparis/status/1790904647630151889

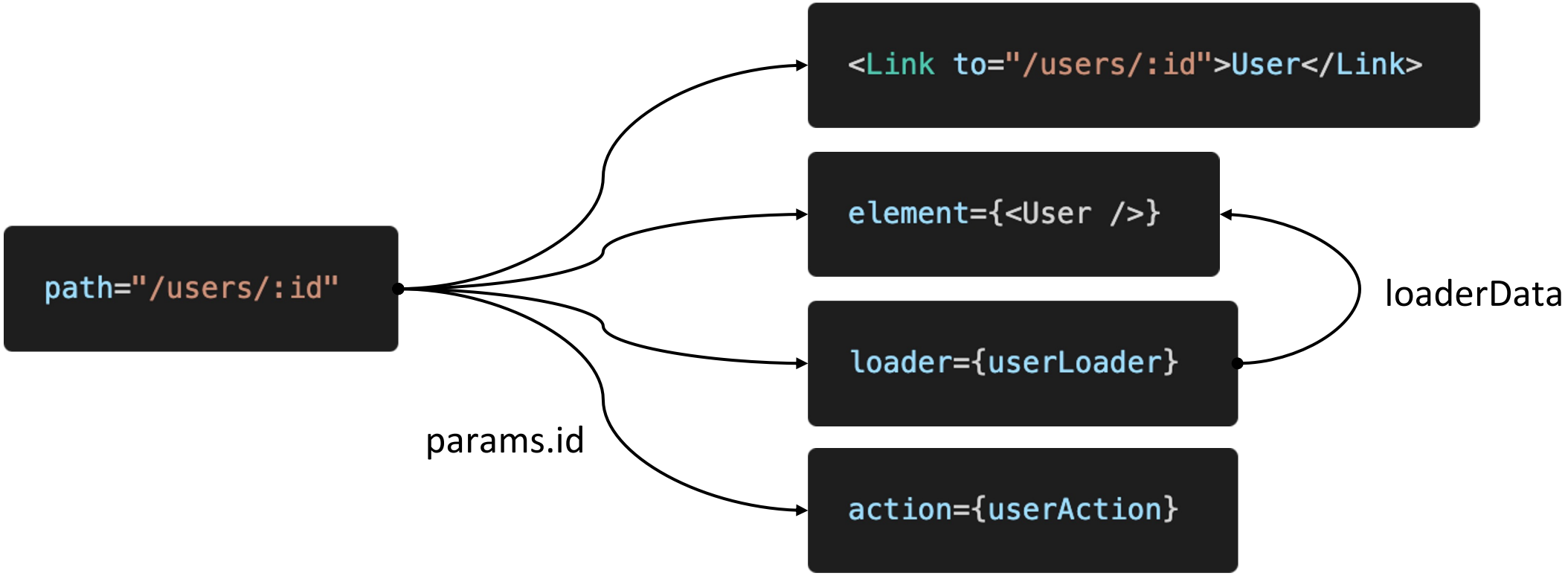
React Router v6.4

- v7の準備としてRemixの概念（loader, action）が追加

```
<Route
  path="/users/:id"
  element={<User />}
  loader={userLoader} // ページ表示前にデータを取得
  action={userAction} // react-routerのFormでPOST時にデータを更新
/>
```

- ページ内で行っていた処理をルーティングの一部として書けるように
- loader, actionはclient / serverどちらにも共通する概念

ルーティング変更時の修正の多さ



React Router v7

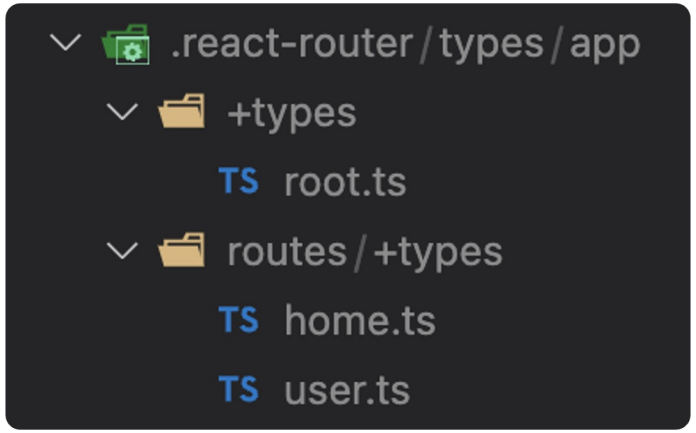
- Viteプラグインによりサーバやビルド周りの機能も入り、Remixと統合
- ページごとのCode Splitting
- 専用のconfig、CLIの導入
- フレームワークとしても使えるように

パスに応じた型の自動生成

パスに応じた型の自動生成

- 準備としてフレームワークとしてのセットアップを行う
- routes.tsにルーティングを書き、**$ react-router typegen** を実行
- .react-router/types下にルート毎の型を生成してくれる

```ts
// app/routes.ts
export default [
  route("/", "routes/home.tsx"),
  route("/users/:id", "routes/user.tsx"),
] satisfies RouteConfig;
```



パスに応じた型の自動生成

- 使う側のルートファイルではcomponentやloaderに生成された型を使う
- 開発モードの場合は裏側で **typegen** が走るため、型は自動更新される

```tsx
// app/routes/user.tsx
import type { Route } from "./+types/user";

export default function User({ params }: Route.ComponentProps) {
  return <p>userId: {params.id}</p>;
}
```

ちなみに中身は

- Info型が肝
- parentsでネストに対応
- Module型は今のファイル

```
// React Router generated types for route:
// routes/user.tsx

import type * as T from "react-router/route-module"

import type { Info as Parent0 } from "../../+types/root.js"

type Module = typeof import("../user.js")

export type Info = {
  parents: [Parent0],
  id: "routes/user"
  file: "routes/user.tsx"
  path: "/users/:id"
  params: {"id": string} & { [key: string]: string | undefined }
  module: Module
  loaderData: T.CreateLoaderData<Module>
  actionData: T.CreateActionData<Module>
}

export namespace Route {
  // ...
}
```

まとめ

まとめ

- ルーティングは少しの工夫で型安全性を高められる
- React RouterはRemixから通信部分の機能を取り込んでフレームワークに
- アップデートでできることが増えているので、頑張りましょう...！

sansan