

RailsからReactを剥がした話





自己紹介

- GMOメディア所属
- 西悠太
- フロントエンドエンジニア
- X: Riya31377928

なぜ移行したのか

ReactからRailsのERBに移行しました。

多くの方が「なぜ？」と思うかもしれません。

RailsからReactに移行するのはよくある話ですが、逆はあまり聞きません。

移行した理由は以下の通りです

- 移行したページはIpなのでReactのメリットがあまりない
- Reactのコードが負債になっている
- 自分以外にReactを書ける人がいない

移行したページはIpなのでReactのメリットがあまりない

移行したページはIpで、 インタラクティブな要素はほとんどありません。

また、 CSRを利用しているのでSEOにもあまりメリットがありません。

そのため、 Reactを使うメリットがあまりないと考えました。

Reactのコードが負債になっている

外部の会社に委託して作成したため、Reactのコードが負債になっていました。
TypeScriptだけど型がない、コンポーネントが分割されていない、コードが複雑、などなど。

自分以外にReactを書ける人がいない

私以外のチームメンバーはRailsをメインで書いているため、Reactを書ける人がいません。
たった数ページのためにReactを書ける人を育成するのは難しいと考えました。

以上の理由から、ReactからRailsのERB移行を決めました。

移行した手順

1. ロジックを整理する

まず始めに、どんな工程を踏んで画面が表示されているのかを整理しました。

画面が表示されるまでの工程は以下の通りです

1. Railsのコントローラーでデータを取得する
2. RailsのコントローラーでReactのコンポーネントにデータを渡す
3. Reactのコンポーネントでデータを加工する
4. 必要に応じてReactからAPIを叩く
5. Reactのコンポーネントでデータを表示する

特にネックになっていたのが、4の「必要に応じてReactからAPIを叩く」でした。
Railsのコントローラーでデータを取得しているのに、Reactから不足分のデータを取得していました。
そのため、何度も再レンダリングが発生していました。

2. コンポーネントを整理する

次に、移行対象のコンポーネントを整理しました。

コンポーネントAはBに依存していて、BはCに依存しているというような複雑な依存関係がありました。

それらを整理し、移行対象のコンポーネントを特定しました。

3. コンポーネントを一個ずつ移行する

コンポーネントを一個ずつ移行していました

1. if文をRubyのコードに変換
2. 共通関数をヘルパーに移植
3. propsをlocal_assignsに変換
4. コードを整理して可読性を上げる
5. JSを可能な限り削除する

苦労した点

見た目の互換性を完璧にReactと合わせないといけない

Ip作成部分がReactで作成されていたため、見た目の互換性を完璧に合わせるのは大変でした。

React独自の機能をバニラJSで実装し直したり、CSSを修正したりしました。

ピクセル単位で見ると誤差はありますが、人間の目では全く分からないレベルになりました。

流れてくるデータが多次元配列

DBから取得したデータが多次元配列で流れていきました。

```
[  
  ["id", 1],  
  ["title", "Title"],  
  ["description", "Description"],  
  ["is_public", true],  
  ["page_url", "hoge"],  
  ["name_param", "hoge"],  
  ["school_id", 1],  
  ["tpl_image", {"id"=>1, "image_url"=>"http://res.cloudinary.com/hoge", "image_smart_url"=>nil}],  
  ...  
]
```

本当はちゃんとしたデータ構造に直したかったのですが、他のページでも参照しているため、今回は直さずに移行しました。

必要なデータには **position** というキーがあるので、それを使ってソートしました。

```
items.map { |item| item.deep_transform_keys(&:to_sym) }.each_with_index.sort_by { |item, index| [item[:position], index] }.map(&:first)  
# mapで配列の中身をシンボルに変換  
# each_with_indexで配列の中身とインデックスを取得  
# sort_byでpositionでソートし、positionが同じ場合はインデックスでソート  
# mapでインデックスを削除
```

if文をRubyのコードに変換

私は普段からReactを書いているので、Rubyの真偽値の扱いに手間取りました。

JSでは、`''` や `0` などの値が `false` として扱われますが、Rubyでは `''` や `0` は `true` として扱われます。

ERBとReactの改行、空白の違い

ERBでは改行や空白がそのまま出力されますが、Reactでは改行や空白が無視されます。
これになかなか気づけず、苦労しました。

ERB

```
<p>
  <%= text %>
</p>
```

```
<p>
  こんにちは
</p>
```

React

```
const Hoge = () => {
  const text = "こんにちは"
  return (
    <p>
      {text}
    </p>
  )
}
```

```
<p>こんにちは</p>
```

コードを整理して可読性を上げる

React側でデータを加工していたため、コードが複雑になっていました。

そのため、コードを整理して可読性を上げるのに苦労しました。

Railsのコントローラーで整形してからデータを渡すように変更したり、必要なデータを一括で取得するように変更したりしました。

バグ検証に時間がかかりましたが、コードが整理されていくのは楽しかったです。

JSを可能な限り削除する

Reactで無駄にJSを書いていた部分があるので、可能な限り削除しました。

例えば、ホバー時にスタイルを変更するという処理は、`onmouseover` と `onmouseout` を使って実装しました。

まとめ

そんなこんなで、ReactからRailsのERBに移行しました。

移行には苦労しましたが、コードが整理されていくのは楽しかったです。

React好きとしては、少し寂しい気持ちもありますが、移行してよかったです。

適材適所で使うことが大切だと思います。

おまけ



<https://zenn.dev/gmomedia/articles/5a6819c5a9e9ad>