

프로그래머스 프론트엔드 아키텍처 변천사

:좋은 개발 경험을 찾아서



Speaker

김은수(Eunsu Kim)

- (현) Grepp(프로그래머스) Frontend Developer
- (전) Mathpresso(콴다) Frontend Developer

프로그래머스

개발자 성장 주기에 맞는 교육, 평가, 채용 서비스를 제공하고 있습니다



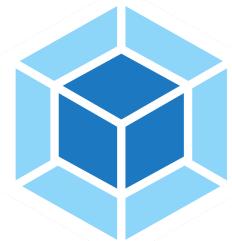
This image shows a mobile device displaying the landing page for the 'Weekly Code Challenge Season 2'. The page features a cartoon bird character holding a pencil, set against a background of rolling hills and clouds. Below the title, there are several orange and yellow achievement icons. At the bottom, there's a navigation bar with tabs for '문제' (Problem), '코드' (Code), and '제출' (Submit). A footer section contains links for '문제' (Problem), '제출' (Submit), and '로그인' (Login).

This image shows a laptop displaying the landing page for developer education. The main headline reads '개발자 채용의 끝, 결국 코드니까'. It features an illustration of a person sitting at a desk with a laptop, surrounded by various programming-related icons like GitHub, Stack Overflow, and Java. Below the main banner, there are two large cards: one for '2020 Dev-Matching' and another for '카카오 경력 개발자 채용 - 프론트엔드 개발 면접'. The footer includes a '제작' (Production) link and a '문제' (Problem) link.

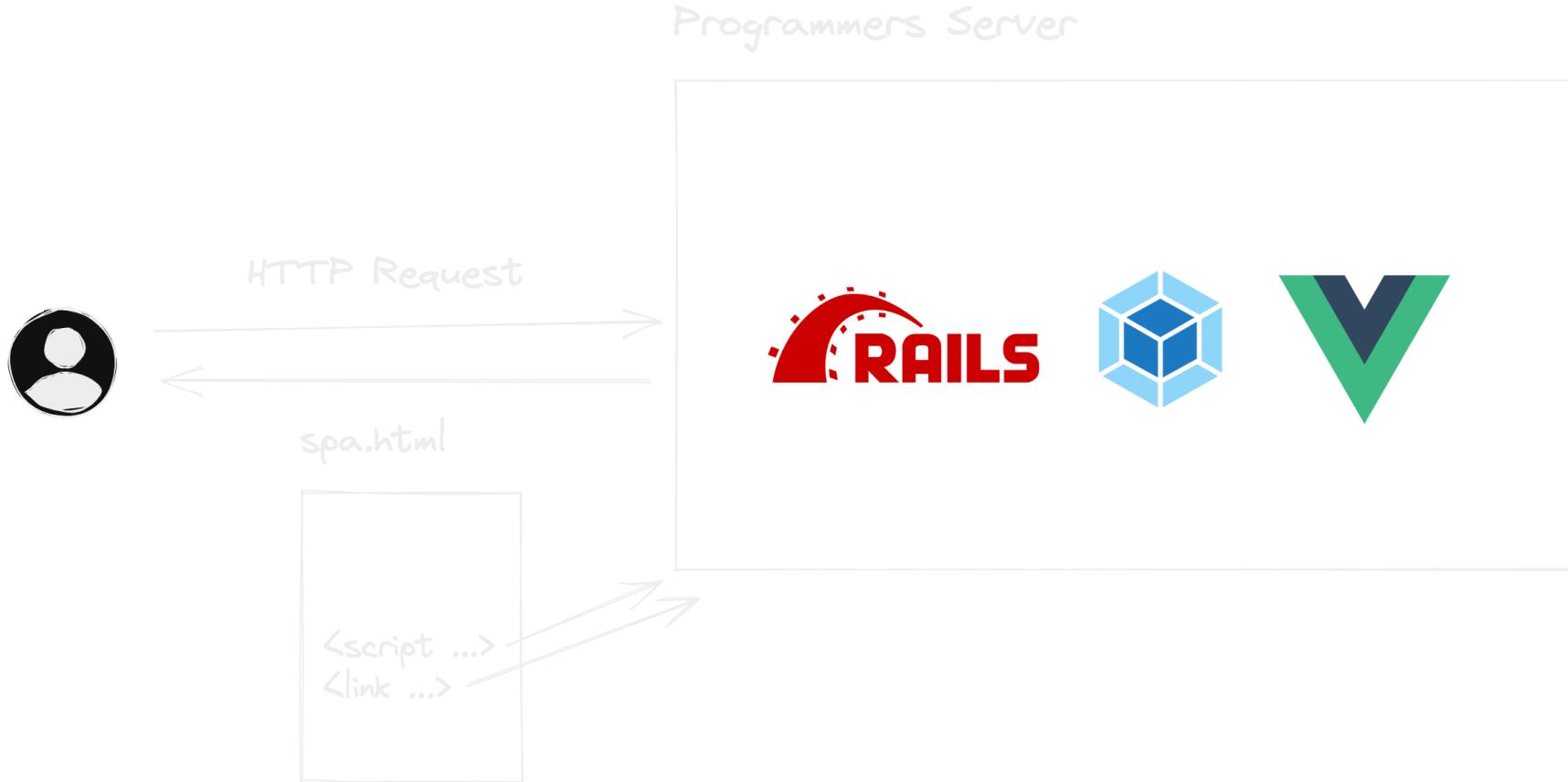
This image shows a desktop browser displaying a list of training courses under the heading '개발자 채용을 위한 디지털 트레이닝'. The list includes several items such as 'K-Digital Training: 프로그래머스 인공지능 대표 프로젝트', 'K-Digital Training: 프로그래머스 사용자형 대표 프로젝트', 'K-Digital Training: 퀄리티 기반 학습 플랫폼 멘토나리팅', and 'K-Digital Training: 퀄리티 기반 학습 플랫폼 멘토나리팅'. Each item has a thumbnail image, a title, and a brief description.

프로그래머스 기술 스택(~2021)

- Ruby on Rails
- Vue(by Webpacker)



프로그래머스 프론트엔드 아키텍처(~2021)



프론트엔드 개발자가 겪는 문제들

- Vue 코드만 수정했는데 배포하는데 너무 오래 걸려요
- 서로 다른 팀이 하나의 package.json을 공유해서 의존성 관리가 힘들어요
- Rails Webpacker에 강한 의존성이 생겼어요

결론적으로, 개발 경험이 안좋아요😭

프론트엔드 개발자가 원하는 것

-  빠르게 개발하고 배포하는 것
-  의존성 관리도 팀별로 알아서 척척
-  Vue말고 React도 쓰고싶다!

그래서 우리는 프론트엔드 아키텍처를 개선하기로 했습니다

New 프론트엔드 아키텍처 설계

-  Do
 - 프론트엔드를 Rails 애플리케이션과 독립적으로 개발/빌드/배포 가능
 - 프론트엔드 코드베이스는 모노레포로 관리
 - 전사 공통 패키지 외에는 팀에서 자유롭게 의존성 도입
 - 기존 아키텍처와 새로운 아키텍처가 공존하고, 점진적(Progressive)으로 마이그레이션 가능
-  Don't
 - 아키텍처를 바꾸느라 제품 개발 프로세스가 멈추면 안됨

선행 조사

프로그래머스가 처한 상황과 비슷한 사례가 있을까?

- Airbnb 사례 - Hypernova
- 오늘의집 사례 - 오늘의집 MSA Phase 1. 프론트엔드 분리작업

Airbnb 사례: Hypernova

A service for server-side rendering your JavaScript views

Server

```
var hypernova = require('hypernova/server');

hypernova({
  devMode: true,
  getComponent(name) {
    if (name === 'MyComponent.js') {
      return require('./app/assets/javascripts/MyComponent.');
    }
    return null;
  },
  port: 3030,
});
```

MyComponent.js

```
const React = require('react');
const renderReact = require('hypernova-react').renderReact;

function MyComponent(props) {
  return <div>Hello, {props.name}!</div>;
}

module.exports = renderReact('MyComponent.js', MyComponent);
```

Airbnb 사례: Hypernova

1. 클라이언트 코드가 서버 코드로부터 분리될 수 있음
2. 점진적으로 적용 가능함

실제로 구현된 사례를 확인함으로써 아키텍처 설계에 대한 힌트를 얻게 되었다!

New 프로그래머스 프론트엔드 아키텍처 - Core Concept

- 프론트엔드 코드를 저장할 모노 레포지토리
- 프론트엔드 애플리케이션 manifest를 Rails 템플릿에 임베딩
 - Webpack Manifest
- Vue/React/Anything in Rails

Webpack Manifest

Webpack으로 빌드된 애플리케이션이 포함하는 코드

1. 개발자가 작성한 소스 코드
2. 작성한 소스 코드가 의존하는 라이브러리 및 벤더(vendor) 코드
3. **Webpack runtime 및 manifest**

Manifest

As the compiler enters, resolves, and maps out your application, it keeps detailed notes on all your modules. This collection of data is called the "Manifest," and it's what the runtime will use to resolve and load modules once they've been bundled and shipped to the browser.

Webpack Manifest Plugin

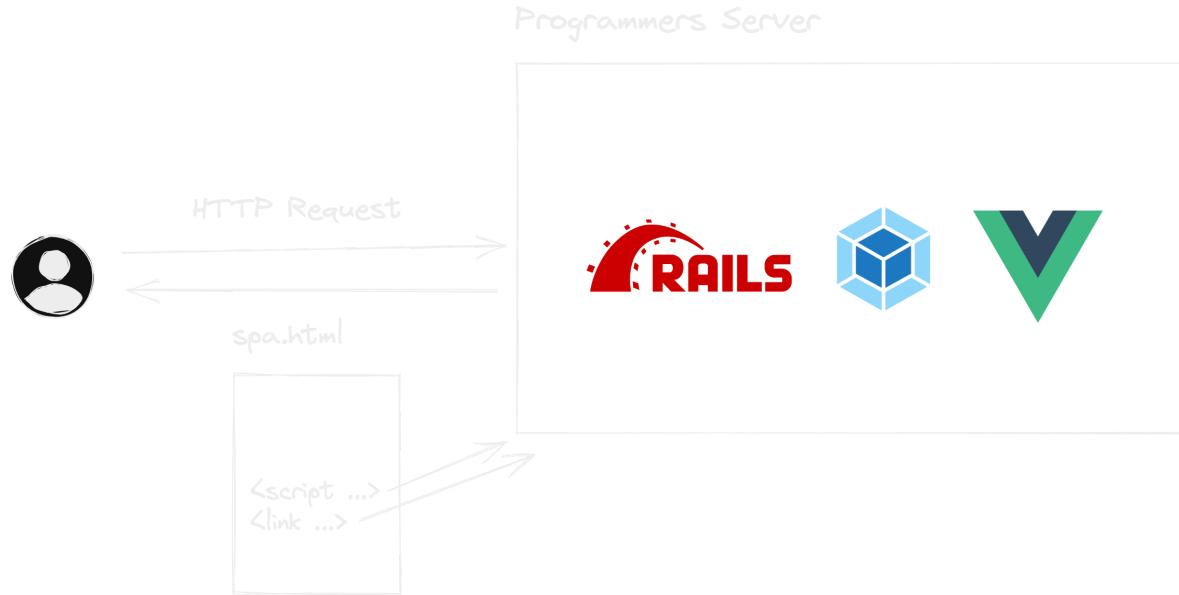
Webpack manifest를 추출할 수 있게 해주는 플러그인

```
// webpack.config.ts
import { WebpackManifestPlugin } from 'webpack-manifest-plugin';

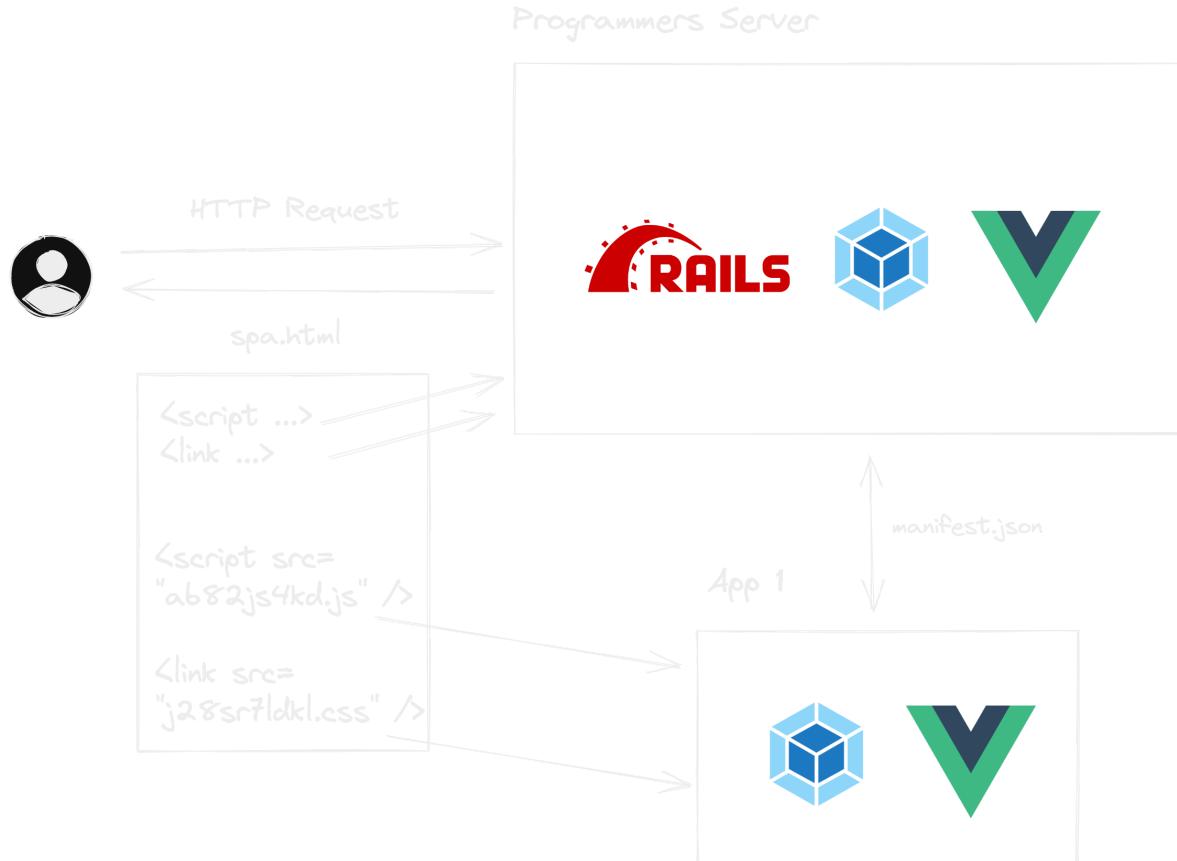
const productionConfig: Configuration = {
  mode: 'production',
  output: { filename: '[contenthash].js' },
  module: { ... },
  plugins: [
    new WebpackManifestPlugin(),
  ],
};

// dist/manifest.json
{
  "main.js": "/b815c1d65ebb736ed574.js",
  "175.css": "/175.67ac9023a9a76244ce98.css",
  "175.css.map": "/175.67ac9023a9a76244ce98.css.map",
  "js": "/02067bc9c4a3d115e4c4.js",
  "js.map": "/02067bc9c4a3d115e4c4.js.map",
  "index.html": "/index.html",
}
```

기존 프로그래머스 프론트엔드 아키텍처

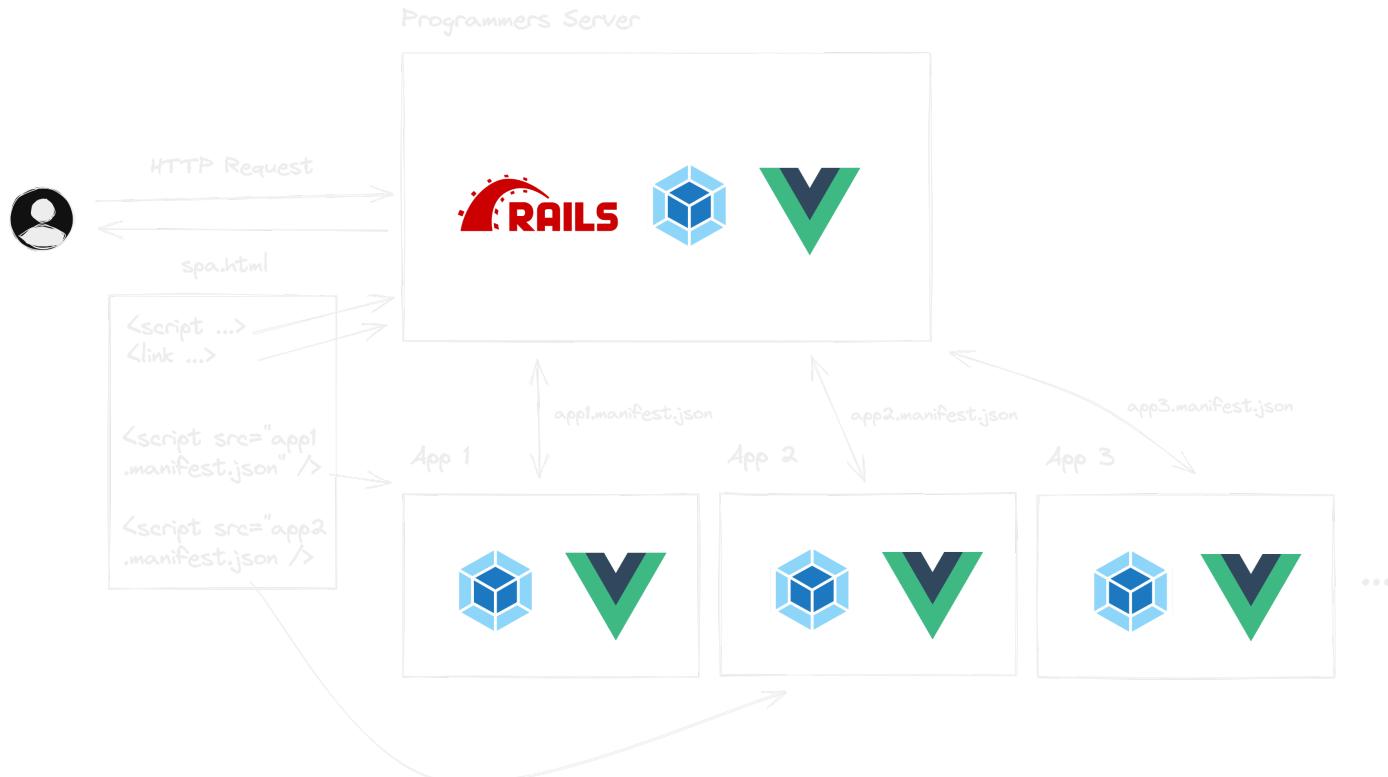


개선된 프로그래머스 프론트엔드 아키텍처



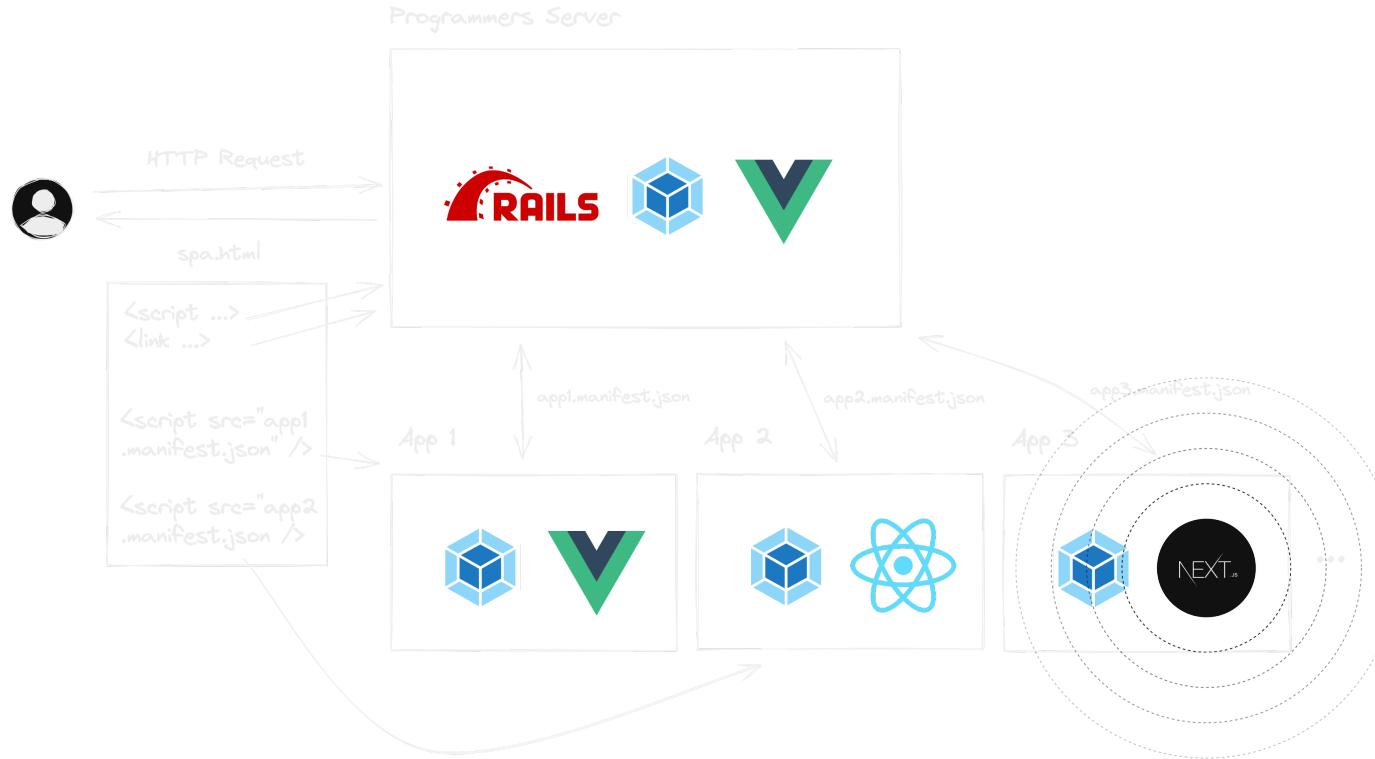
개선된 아키텍처의 장점 - (1) Scalability

기존 Rails 코드베이스의 수정 없이 새로운 웹 애플리케이션 개발 및 배포가 가능



개선된 아키텍처의 장점 - (2) Independency

애플리케이션의 의존성을 독립적으로 관리가 가능

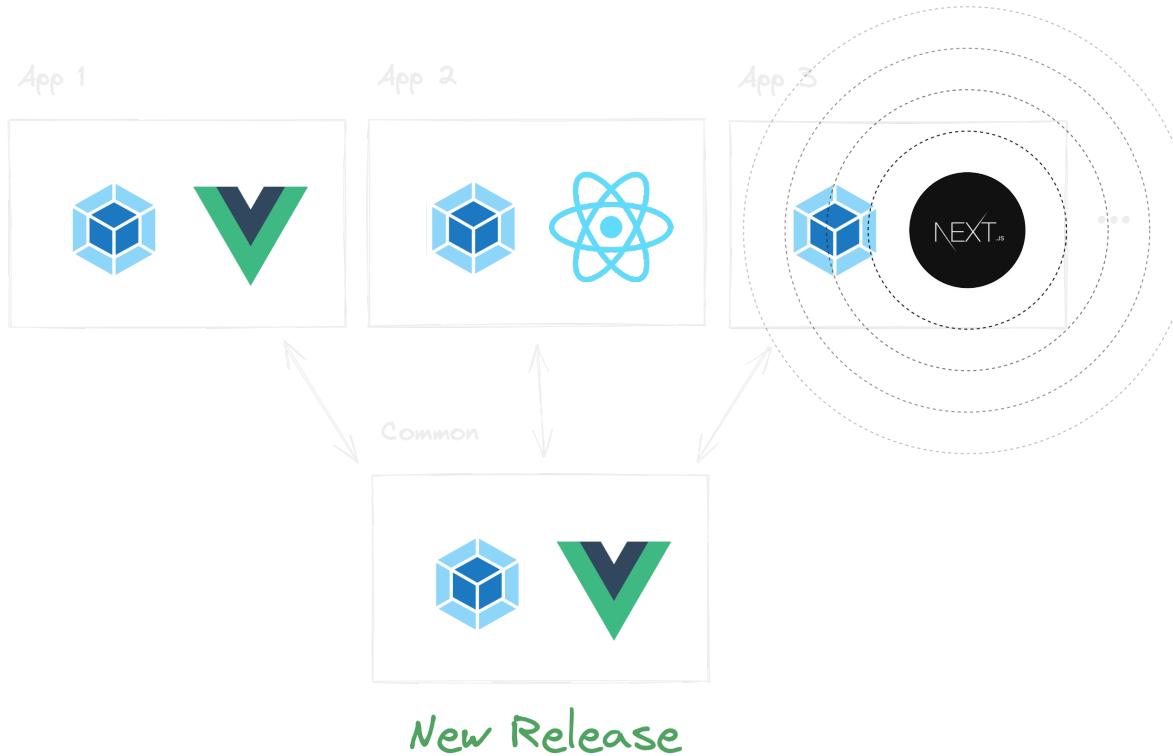


그렇게 프로그래머스 프론트엔드 개발자들은 행복하게 잘 살았습니다

그렇게 프로그래머스 프론트엔드 개발자들은 행복하게 잘 살았습니다
…가 또 불편한 점이 생겼습니다

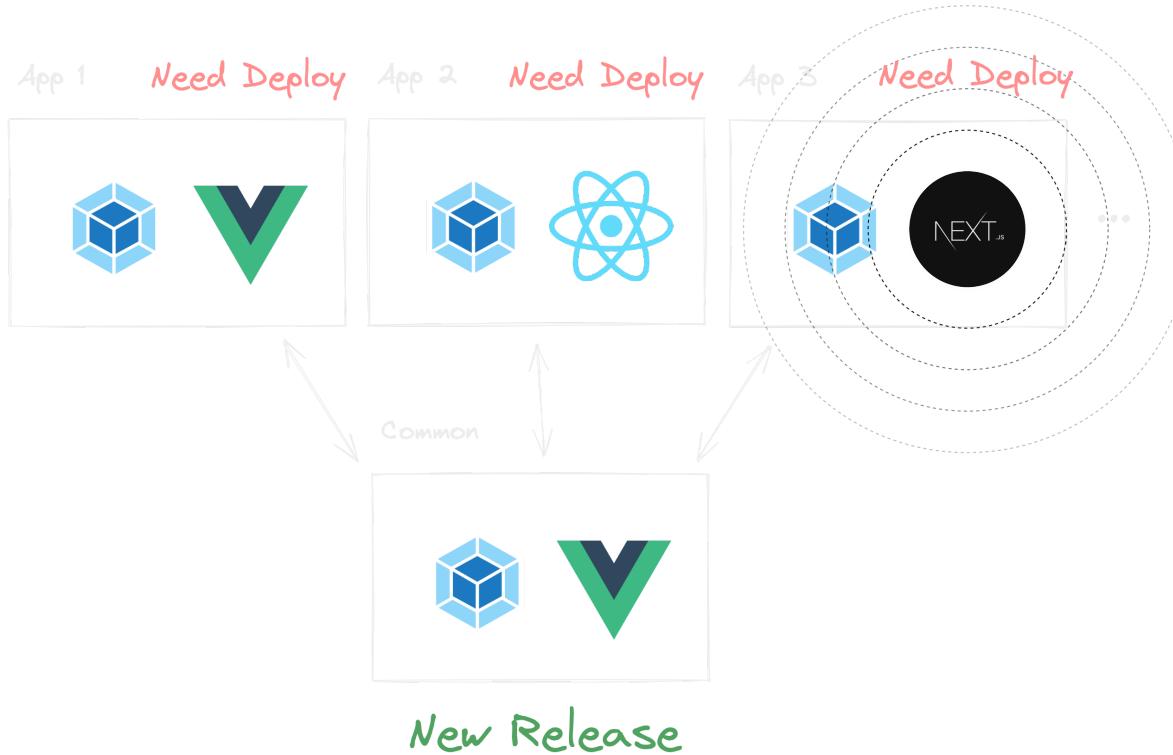
개선된 아키텍처의 불편한 점

각 애플리케이션에서 의존하는 코드가 새로 릴리즈되면 해당 사항을 반영하기 위해 모든 앱을 다시 배포해야 함



개선된 아키텍처의 불편한 점

각 애플리케이션에서 의존하는 코드가 새로 릴리즈되면 해당 사항을 반영하기 위해 모든 앱을 다시 배포해야 함



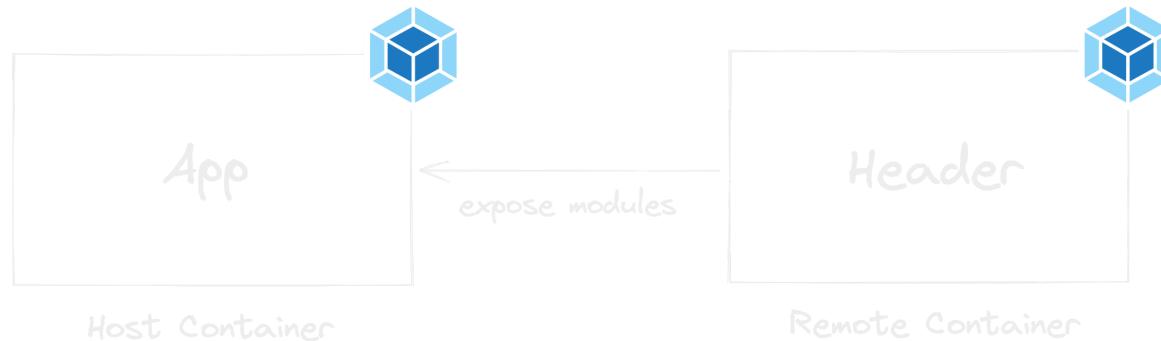
앱을 다시 배포하지 않고 의존하는 코드의 변경사항을 런타임에 바로 적용할 수 없을까?

앱을 다시 배포하지 않고 의존하는 코드의 변경사항을 런타임에 바로 적용할 수 없을까?

Webpack 5 Module Federation

Webpack 5 Module Federation - Core Concept

- 빌드된 애플리케이션 = 컨테이너
 - 호스트(Host) 컨테이너: 외부에 노출된 모듈을 불러와 사용
 - 리모트(Remote) 컨테이너: 모듈을 외부로 노출(expose)
- Webpack 5버전 부터 포함된 Container Plugin을 사용하여 적용 가능
 - Container Plugin을 추상화 한 Module Federation Plugin을 사용하면 더 쉽게 적용 가능



Module Federation Plugin

Host 컨테이너의 webpack config

```
import { container } from 'webpack';

const config: Configuration = {
  // ...생략
  plugins: [
    new container.ModuleFederationPlugin({
      remotes: {
        remoteApp:
          `remoteApp@${
            process.env.APP_URL
          }/remoteEntry.js`,
      },
    }),
  ],
}
```

Remote 컨테이너의 webpack config

```
import { container } from 'webpack';

const config: Configuration = {
  // ... 생략
  plugins: [
    new container.ModuleFederationPlugin({
      name: 'remoteApp',
      filename: 'remoteEntry.js',
      exposes: {
        './Header': './src/components/Header.tsx',
      },
    }),
  ],
};
```

Module Federation 사용 예시

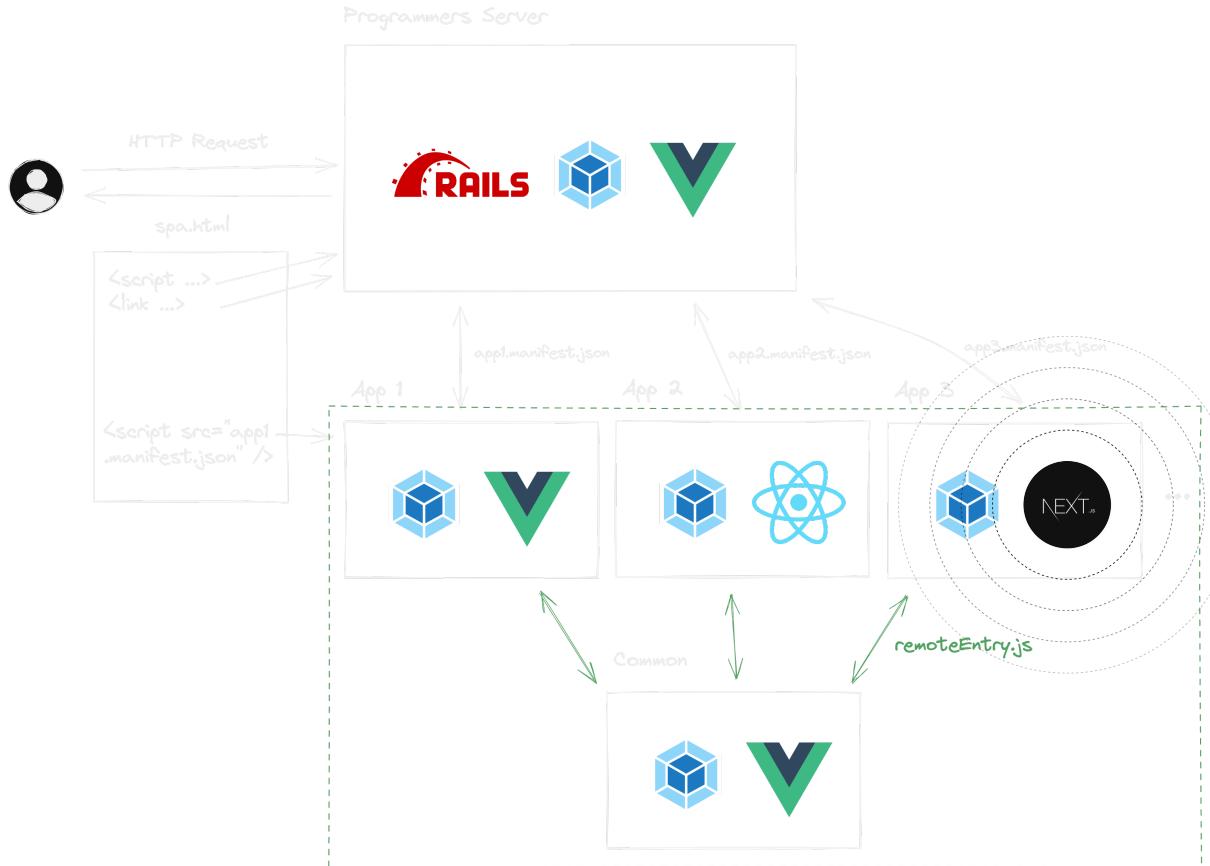
```
import { LoadingSpinner } from '@/components>LoadingSpinner'
import { Suspense } from 'react';

const Header = lazy(() => import('remoteApp/Header'));

function CustomHeader() {
  return (
    <Suspense fallback={<LoadingSpinner />} >
      <Header />
    </Suspense>
  );
}

export default CustomHeader;
```

개선된 프론트엔드 아키텍처(현재)



그래서 우리는 얼마나 빠르게 제품을 개발하고 배포할 수 있게 되었을까요?

프론트엔드 프로덕션 배포 소요 시간

기존: 30 ~ 40분

현재: < 5분

프로덕션 릴리즈 횟수

2022년 3월: 46회

2022년 11월: 173회

Closed Pull Request 개수

기존 repo: 6년 4개월 간 5500여개 (72 PRs / month)

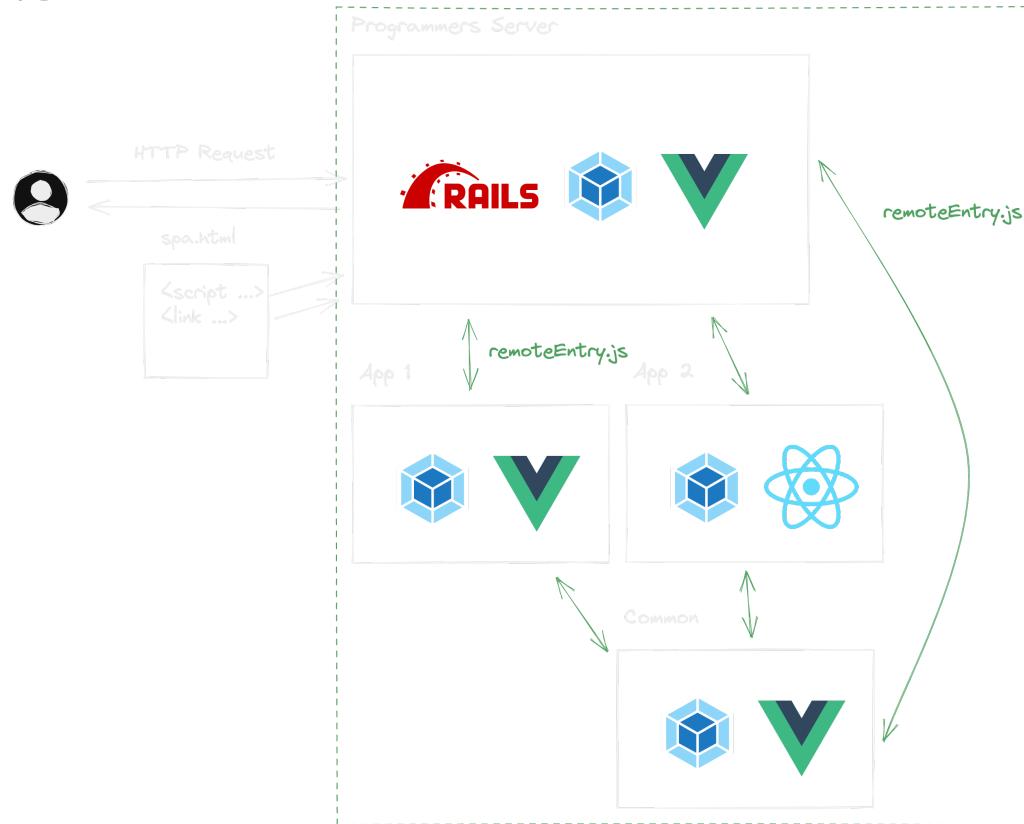
신규 repo: 8개월 간 1400여개 (175 PRs / month)

Next Steps

- Rails 번들러 마이그레이션
- Rails에서 번들러 의존성을 제거
- 개발 경험 개선

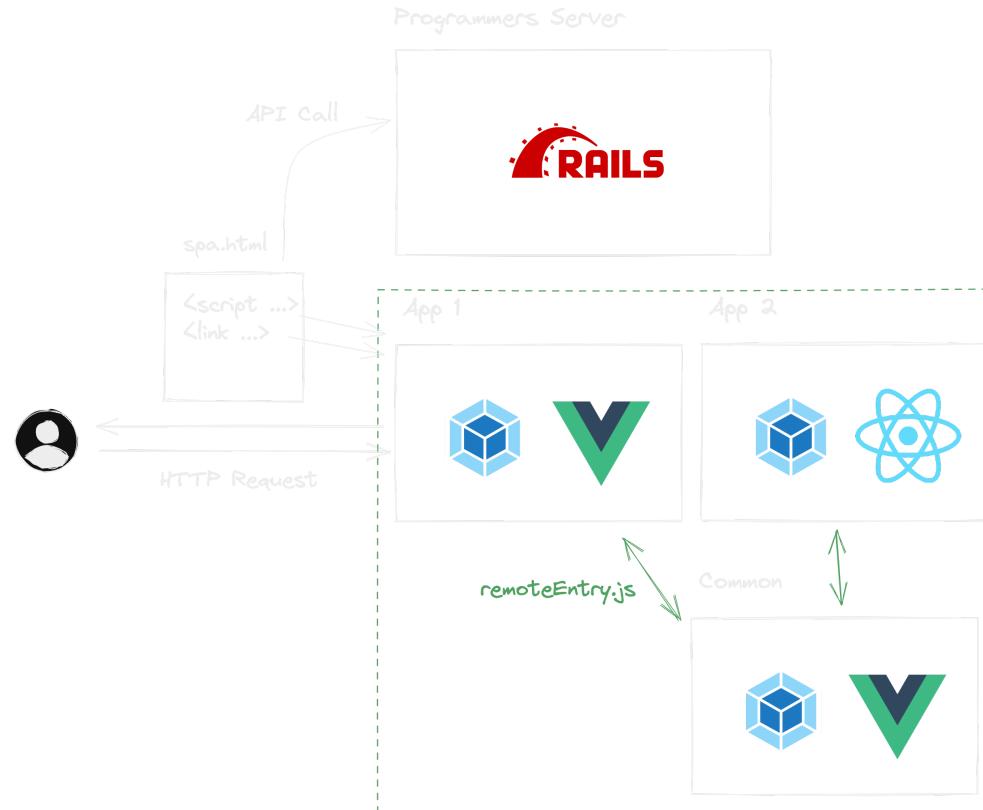
Next Steps - (1) Rails 번들러 마이그레이션

Rails Webpacker를 Shakapacker로 마이그레이션하여 Webpack 5버전을 사용할 뿐만 아니라 babel 외의 빌드 도구 (swc, esbuild)도 사용 가능



Next Steps - (2) Rails에서 번들러 의존성을 제거

Rails로부터 번들러(Webpacker) 의존성을 분리함으로써 Rails는 API 서버처럼 동작



Next Steps - (3) 개발 경험 개선

- Remote module의 type checking
- 앱 간의 커뮤니케이션 컨벤션 정의
- Best practice 탐구

프로그래머스의 프론트엔드 개발 경험을 개선하기 위한 모험은 계속됩니다

Recap

- Rails와 Vue가 함께 있는 레포지토리에서 프론트엔드 개발 경험이 좋지 않음
- 프론트엔드 코드베이스를 Rails 코드와 분리시켜 독립적으로 개발/빌드/배포 할 수 있고 점진적으로 마이그레이션 가능한 아키텍처를 설계
 - Airbnb의 Hypernova 사례를 바탕으로
 - 신규 프론트엔드 애플리케이션의 Webpack manifest를 Rails 템플릿에 임베딩
 - 신규 프론트엔드 애플리케이션 간의 Module Federation을 통한 Runtime Integration
- 개발 경험을 개선하기 위한 노력이 여전히 진행 중
 - Rails 번들러 마이그레이션
 - Rails에서 번들러 의존성 제거
 - 신규 프론트엔드 코드베이스의 개발 경험 개선