우리은행

MFE란 무엇인가?

거대한 서비스를 조립식으로 관리하는 프론트엔드 설계 전략

MFE (Mad Four Elites)팀

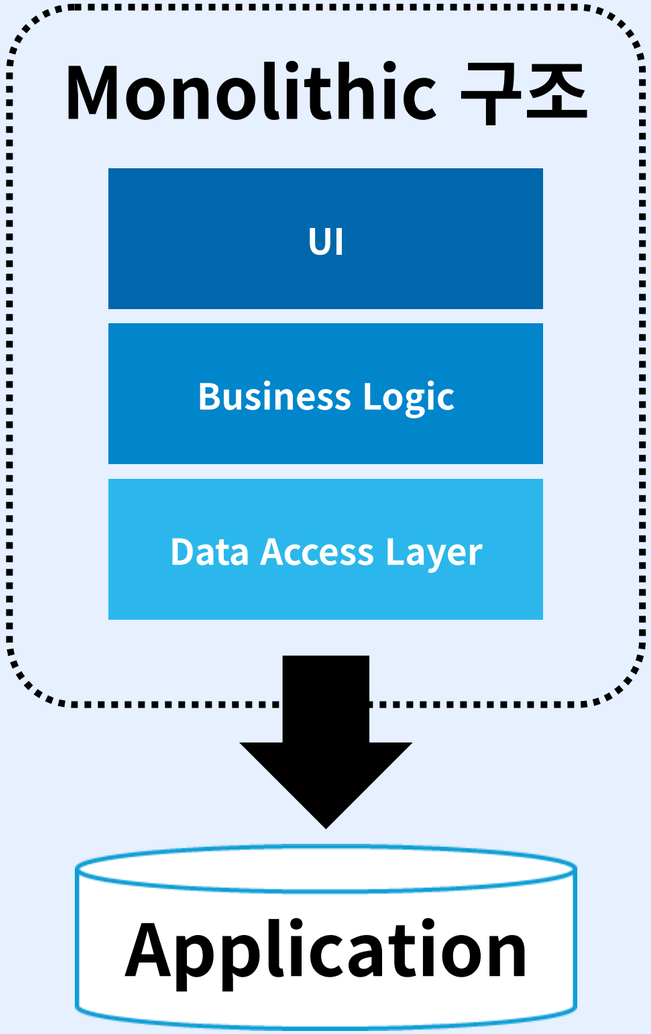
이유림 박주호 송민혁 이건희

우리은행

목차

1. MFE Architecture란 무엇인가?
2. MFE Architecture 장단점
3. MFE 관리 방식 - 모노레포 VS 멀티레포
4. MFE 구현 방식 1 - iframe
5. MFE 구현 방식 2 - Module Federation

1. MFE란 무엇인가? - MFE 등장 배경



하나의 통합된 코드베이스로 여러 비즈니스 기능을 수행하는 전통적인 아키텍처 스타일

단일 Application 내에 서비스의 모든 로직이 들어가 있는 구조

출처 : 두두네 클라우드

1. MFE란 무엇인가? - MFE 등장 배경

Monolithic Architecture의 장점은?

- 빠른 개발
- 배포 용이성
- 낮은 초기 비용
- 유지보수 용이성

1. MFE란 무엇인가? - MFE 등장 배경

Monolithic Architecture의 단점은?

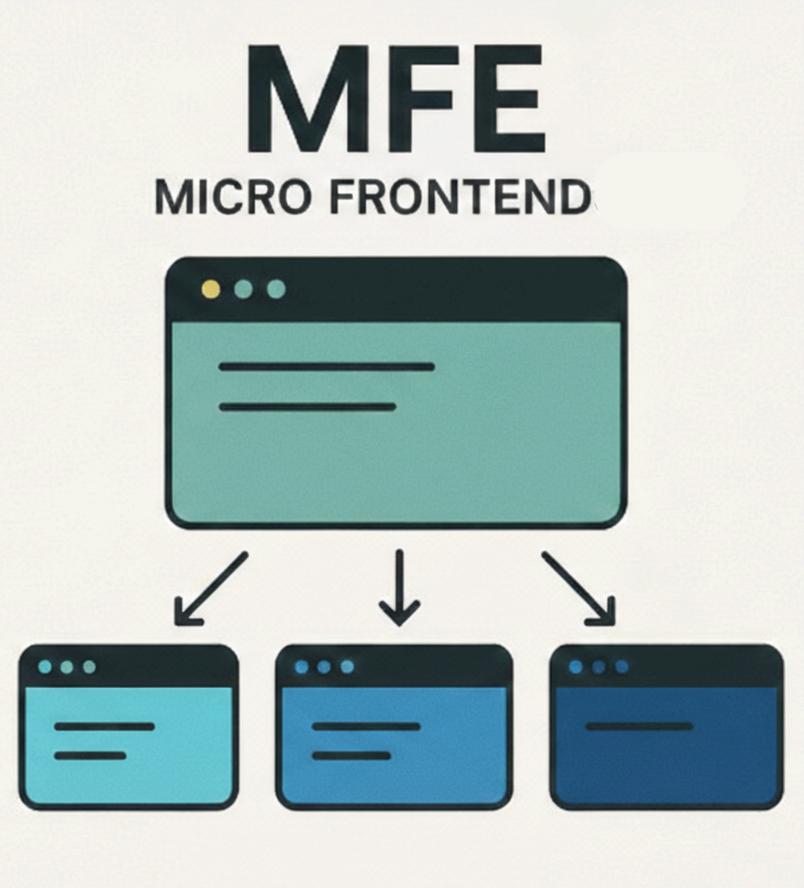
- 배포 및 빌드 속도 저하
- 유지보수 및 복잡도 증가
- 전체 시스템 위험
- 협업의 어려움

1. MFE란 무엇인가?

그렇다면 Monolithic 구조의 단점을 해결하기 위한 방법은?

MFE
(Micro Front-End)

1. MFE란 무엇인가?



하나의 대규모 Front-End Application을 여러 개의 작고 독립적인 Application으로 분리한 Architecture Pattern

목적

- 독립적인 배포
- 팀 단위 개발 환경 형성
- 유지보수 효율성 향상

출처 : Gemini

2. MFE Architecture의 장단점

Micro Front-End의 장점은?

- 독립적인 배포 가능
- 기술 사용의 유연성
- 스케일링 용이성
- 리스크 분산 처리

2. MFE Architecture의 장단점

Micro Front-End의 단점은?

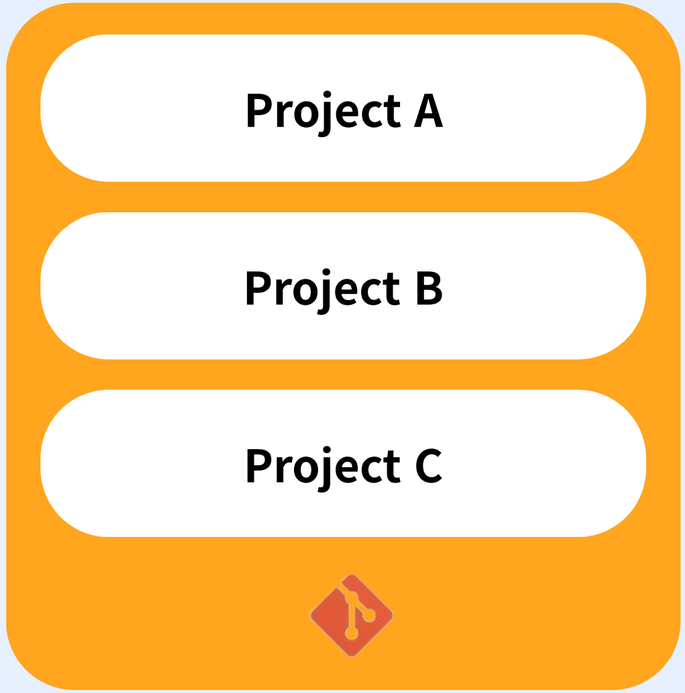
- 운영 복잡성 증가
- 성능 저하
- 일관성 유지의 어려움
- 데이터 불일치

2. MFE Architecture의 장단점

Micro Front-End의 사용 시 고려사항?

- 사용 기술스택 관리
- UI/UX 일관성 유지
- 독립적인 배포 파이프라인
- 상태 관리

3. MFE 관리 방법 - 모노레포(Mono-Repo)



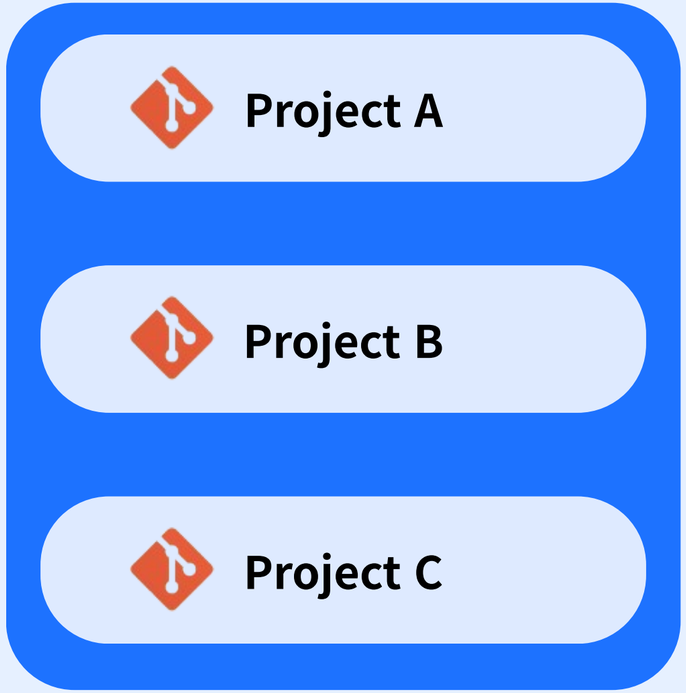
Mono-Repo

여러 프로젝트, 라이브러리, 앱을 단일 저장소에서 통합 관리하는 방식

장점	단점
• 소스코드의 무결성 보장 • 통합된 버전 관리 가능 • 코드 공유 및 재사용 용이 • 쉬운 의존성 관리	• 무분별한 의존성 연결 • 형상 관리 및 CI 속도 저하 • 긴 빌드 시간

출처 : Fast Campus, Git

3. MFE 관리 방법 - 멀티레포(Multi-Repo)



Multi-Repo

프로젝트, 서비스 또는 모듈별로 각각 독립된 코드 저장소를 만들어 관리하는 방식

장점	단점
• 유지보수 용이 • 빠른 배포 속도 • 효율적인 테스트 수행 • 의존성 관리 용이	• 높은 코드 중복 가능성 • 일관성 유지 어려움 • 서비스 간 버전 충돌

출처 : Fast Campus, Git

우리은행

MFE 구조를 반드시 사용 X

개발 상황에 대한 이해가 우선

MFE 구현 방식 1

iframe

4. iframe 방식 - 개념 정의

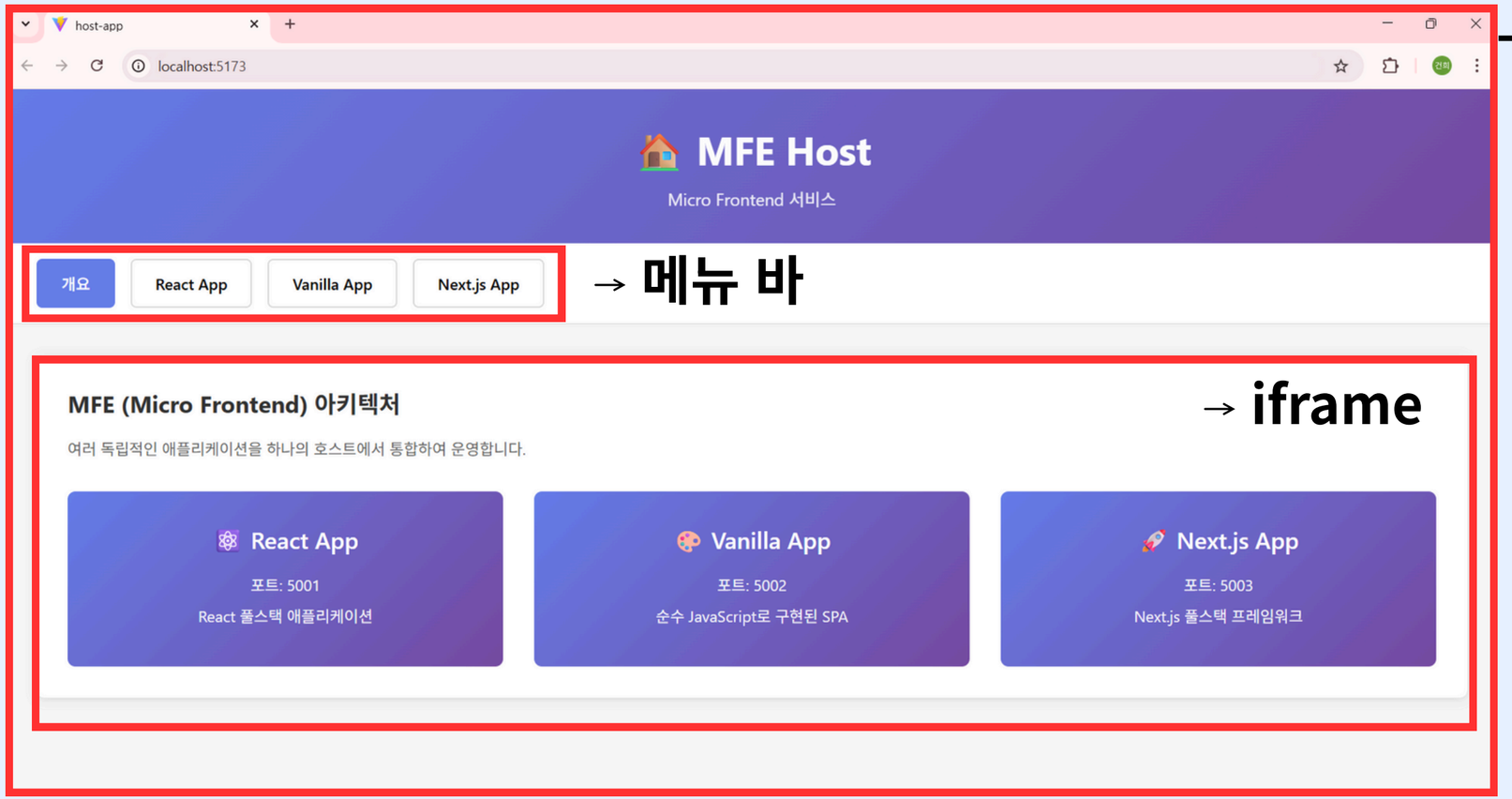
하나의 HOST 웹 페이지 안에 다른 HTML 문서를 삽입하는 요소

- 단순, 전통적 통합 전략
- 문서 단위 통합
- 별도의 browsing context에서 독립 실행

“코드 공유가 아닌 문서를 불러오는 것”

4. iframe - 동작 과정

4-1. HOST 화면



→ HOST페이지

→ 메뉴 바

→ iframe

http://localhost:5173

4. iframe - 동작 과정

4-2. 클릭 시 상태 변경

```jsx
<button
  className={activeTab === 'react' ? 'active' : ''}
  onClick={() => setActiveTab('react')}
>
  React App
</button>
```

```jsx
<button
  className={activeTab === 'nextjs' ? 'active' : ''}
  onClick={() => setActiveTab('nextjs')}
>
  Next.js App
</button>
```

host-app/App.jsx

4. iframe - 동작 과정

4-3. iframe DOM 생성

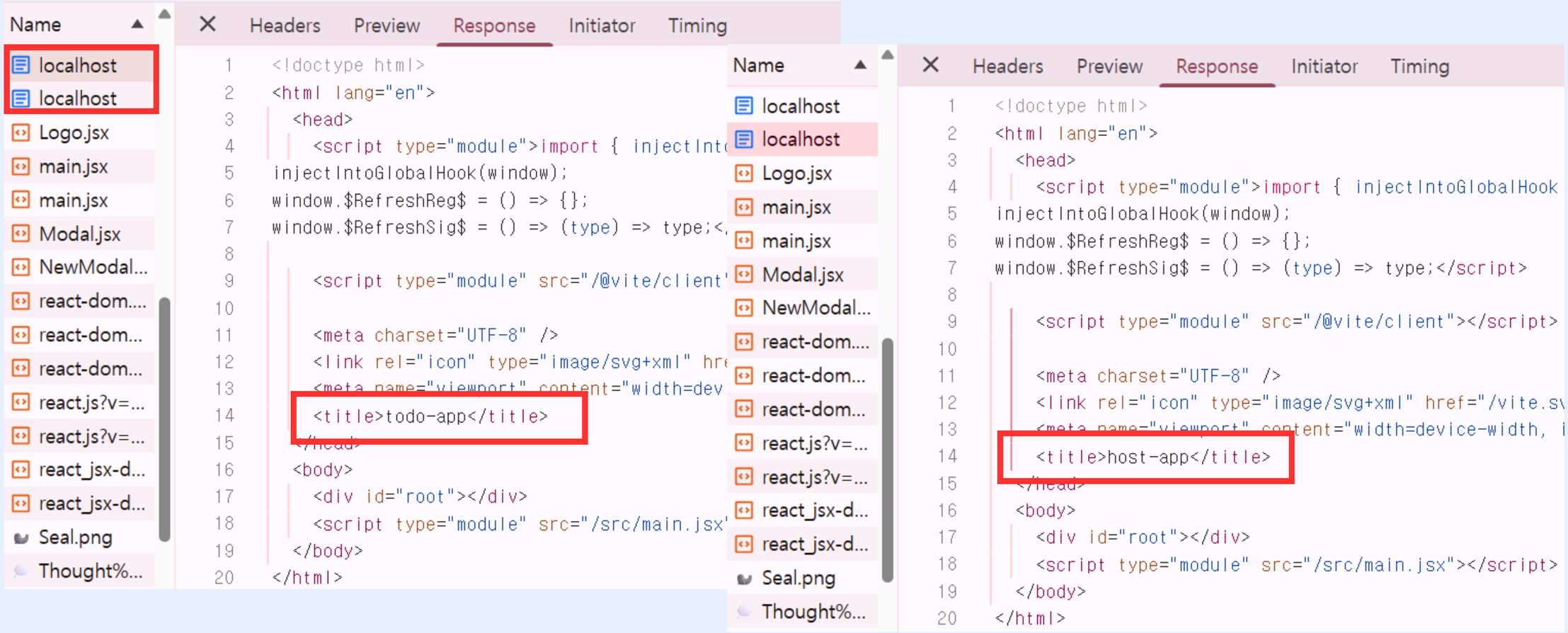
```jsx
{activeTab === 'react' && (
  <section className="app-section">
    <h2>⚛️ React App</h2>
    <div className="iframe-container">
      <iframe
        src="http://localhost:5001"
        title="React App"
        className="remote-iframe"
      />
    </div>
  </section>
)}
```

```jsx
{activeTab === 'nextjs' && (
  <section className="app-section">
    <h2>🚀 Next.js App</h2>
    <div className="iframe-container">
      <iframe
        src="http://localhost:5003"
        title="Next.js App"
        className="remote-iframe"
      />
    </div>
  </section>
)}
```

host-app/App.jsx

4. iframe - 동작 과정

4-4. HTTP 요청



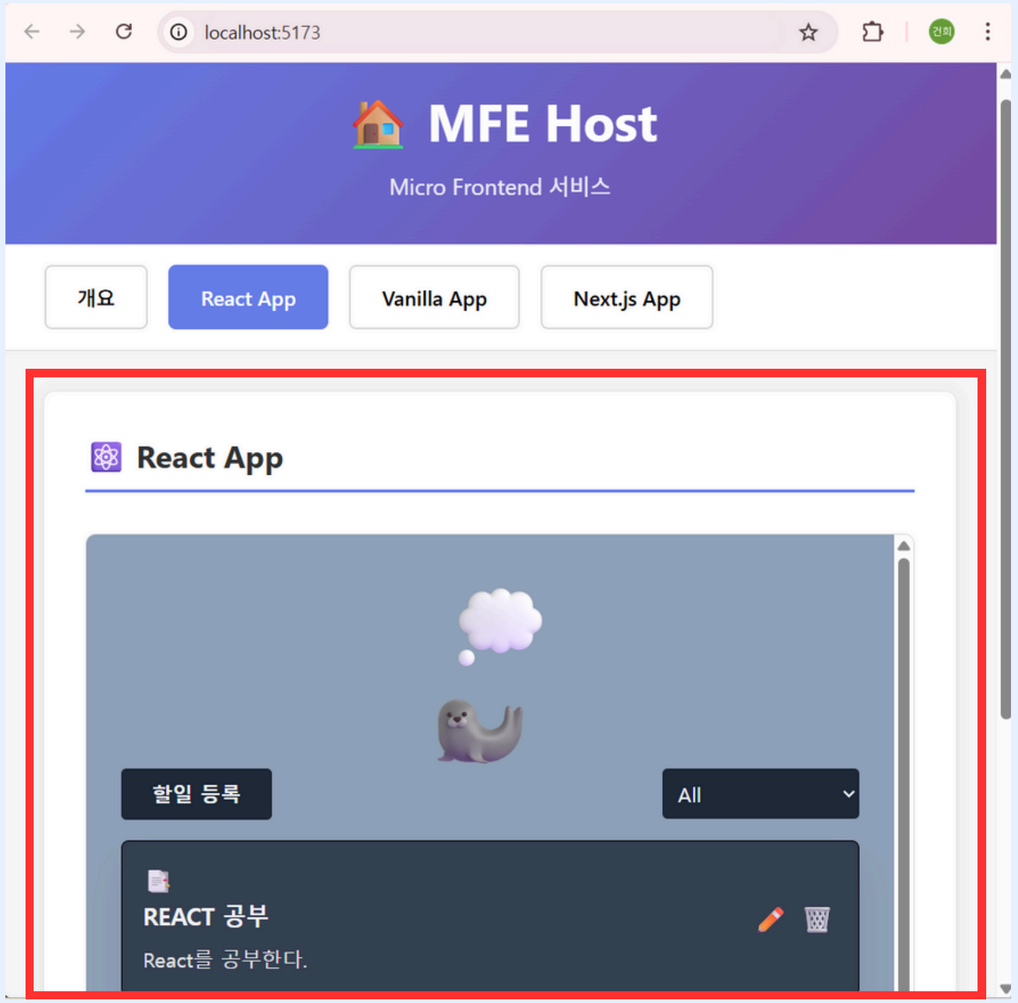
모듈 import X

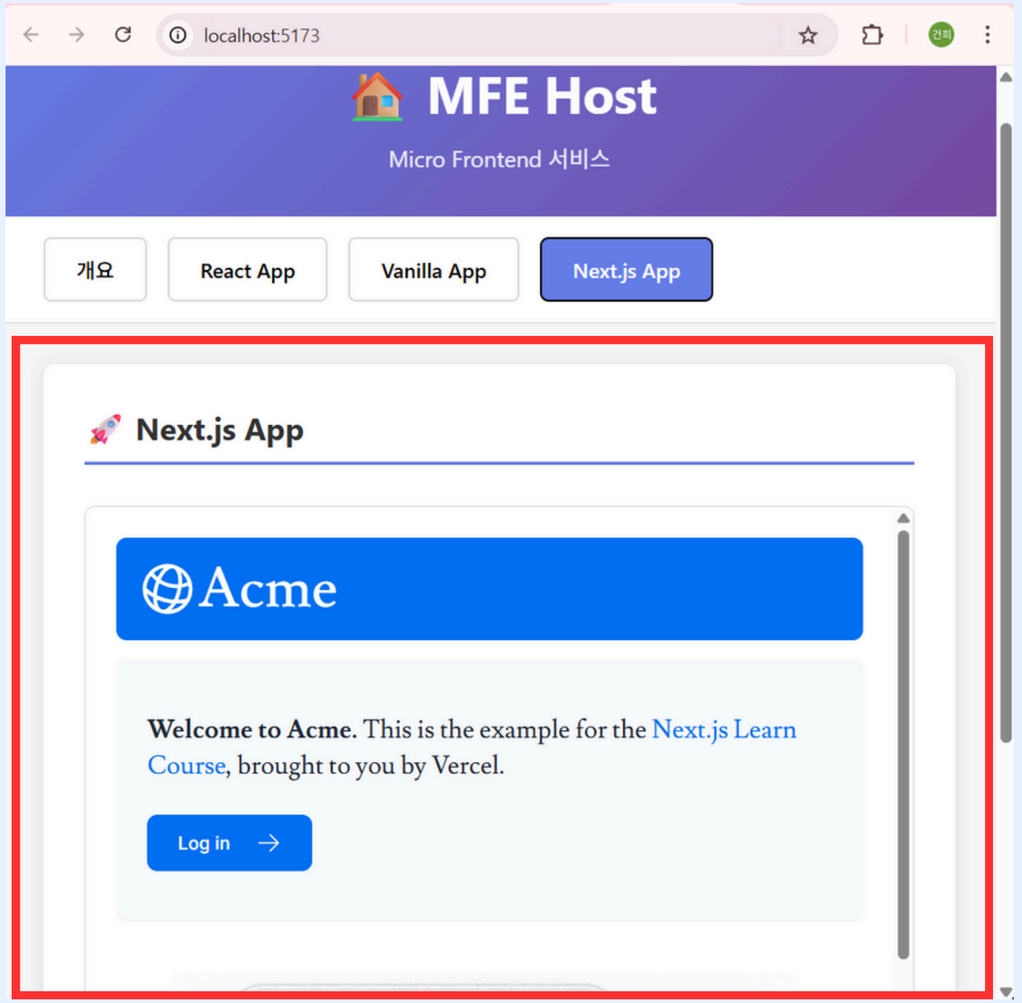
HTML 문서 요청 O

이미지 : 호스트앱 개발자 도구

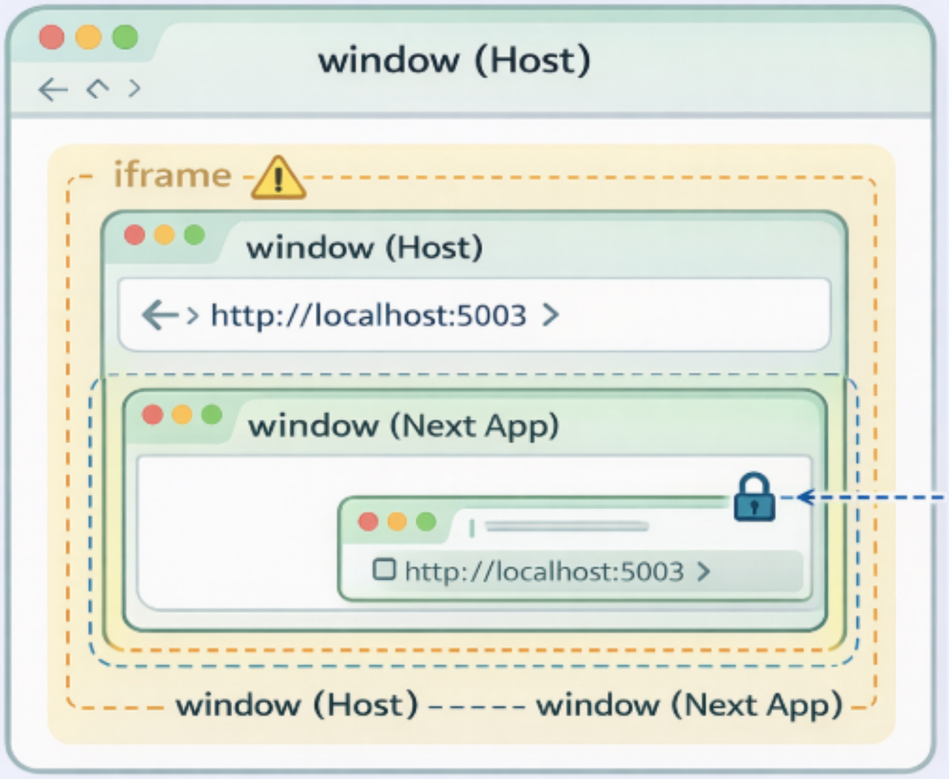
4. iframe - 동작 과정

4-5. 페이지 렌더링





4. iframe - 구조



HOST와 완전히 격리된 환경

서로의 DOM에 직접 접근 불가
JS 변수 공유 X
CSS 충돌 발생 X

통신이 필요한 경우? postMessage API에 의존

! nextjs-app / react-app 에서는 아무 설정 X

독립적 SPA 페이지 / HOST의 iframe에 URL을 포함하여 요청

이미지 : Chat GPT 생성

MFE 구현 방식 2

Module Federation

5. 기술 시연 및 코드 설명 - MF란 무엇인가?

Module Federation

하나의 앱을 독립적인 배포가 가능한 모듈 단위 (청크)로 나누어
브라우저 런타임에 통합하는 방식

JS를 통한 런타임 통합

5. 기술 시연 및 코드 설명 - 용어 설명

HOST

원격 모듈들을 로드하여 통합하는
주 어플리케이션

Remote

원격 컨테이너로부터
런타임에 로드되는 모듈

5. 기술 시연 및 코드 설명 - MF 동작과정

Host-Remote 관계 설정

Remote-app
vite.config.js

```
plugins: [
  federation({
    name: 'vanilla_app',
    filename: 'remoteEntry.js',
    exposes: {
      './bootstrap': './js/bootstrap.js',
    },
```

Host-app
vite.config.js

```
plugins: [react(),
  federation({
    name: 'host_app',
    remotes: {
      remoteVanilla: 'http://localhost:5002/assets/remoteEntry.js',
    },
```

5. 기술 시연 및 코드 설명 - MF 동작과정

remote_app/remoteEntry.js

```
// remote-app/dist/assets/remoteEntry.js
let moduleMap = {
  "./bootstrap": () => {
    return __federation_import('./__federation_expose_Bootstrap-RXtAXZgd.js')
      .then(module => ... )
  },
};
```

런타임에 필요한 모듈을 찾아가 로드할 수 있게 해주는 엔트리 포인트

5. 기술 시연 및 코드 설명 - MF 동작과정

Host_app/App.jsx

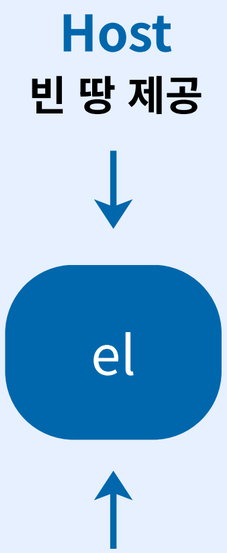
```
import('remoteVanilla/bootstrap').then((module) => {
  const { mount } = module.default || module
  if (vanillaContainerRef.current) {
    const unmountFn = mount(vanillaContainerRef.current)
    unmountFuncsRef.current.vanilla = unmountFn
    setLoading(prev => ({ ...prev, vanilla: false }))
  }
})
```

import 실행되는 순간 브라우저는 remote app의 코드를 실시간으로 로드

5. 기술 시연 및 코드 설명 - MF 동작과정

Remote_app/bootstrap.js

1. UI 주입 : mount



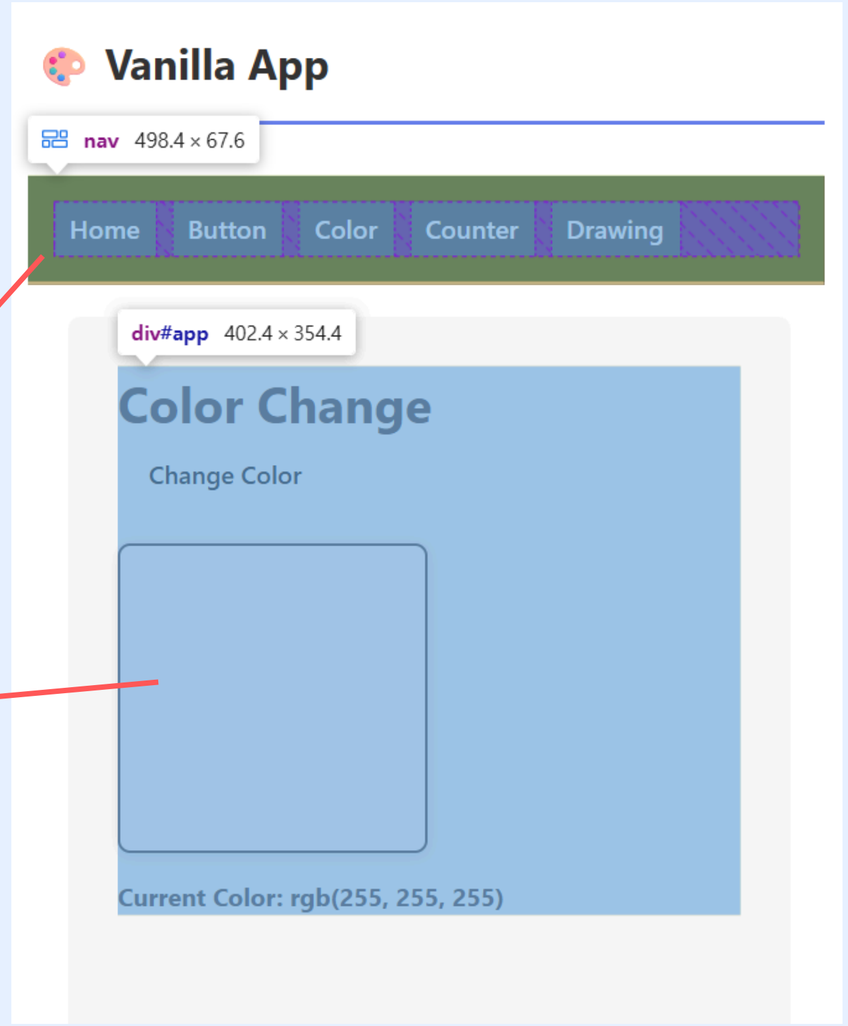

```
// el: Host가 넘겨준 빈 div
const mount = (el) => {
if (!el) return;

// 1. 네비게이션(nav) 생성
const nav = document.createElement('nav');
nav.style.cssText = 'padding: 1rem; background: #333; ...';

// 2. 컨테이너 및 실제 콘텐츠 영역 생성
const appContainer = document.createElement('div');
const content = document.createElement('div');
content.id = 'app';

appContainer.appendChild(content);

// 3. Host DOM에 주입
el.appendChild(nav);
el.appendChild(appContainer);
}
```

5. 기술 시연 및 코드 설명 - MF 동작과정

Remote_app/bootstrap.js

1. UI 주입 : mount

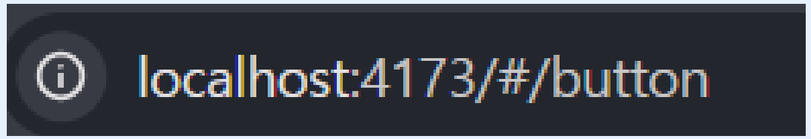
```
<section class="app-section">
  <h2> Vanilla App</h2>
  <div>
    <nav style="padding: 1rem; background: rgb(51, 51, 51); border-bot
    tom: 2px solid rgb(102, 102, 102); display: flex; gap: 10px;">
    </nav> flex
    <div style="min-height: 500px; padding: 2rem; background: rgb(249,
    249, 249); border-radius: 8px; margin: 1rem; color: rgb(51, 51, 5
    1);">
      <div id="app"> == $0
        <h1>Color Change</h1>
        <button id="colorBtn">Change Color</button>
        <div id="colorPreview" style="margin-top: 1.5rem;"></div>
      </div>
```



5. 기술 시연 및 코드 설명 - MF 동작과정

Remote_app/bootstrap.js

2. 내부 라우팅 (해시 라우터)




```
const handleNavigation = () => {
    // 주소창의 '#' 뒤만 읽어서 어떤 페이지를 보여줄지 결정 (Host URL 보호)
    const path = window.location.hash.slice(1) || '/';
    const page = routes[path] || Home;
    content.innerHTML = page(); // 화면 전환
};
```

 # 뒤의 값이 변경되더라도 새로고침 없이 문서 내부 상태 변경으로 처리

→ Remote 앱 내부에서만 페이지 전환 가능

5. 기술 시연 및 코드 설명 - MF 동작과정

Remote_app/bootstrap.js

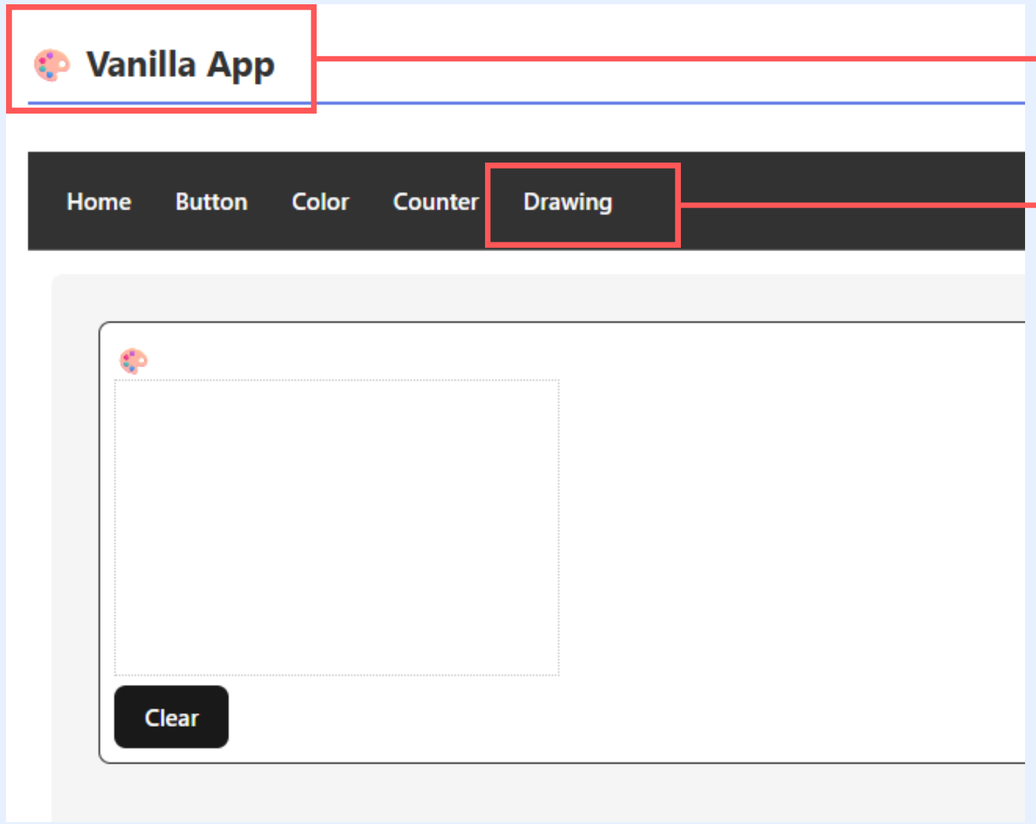
3. Clean up

사용자가 다른 탭으로 이동 할 때, 리스너 제거 및 DOM 정리

```
  return () => {
    window.removeEventListener('hashchange', handleNavigation);
    el.innerHTML = '';
  };
};
```

5. 기술 시연 및 코드 설명 - MF 동작과정

Host이자 Remote



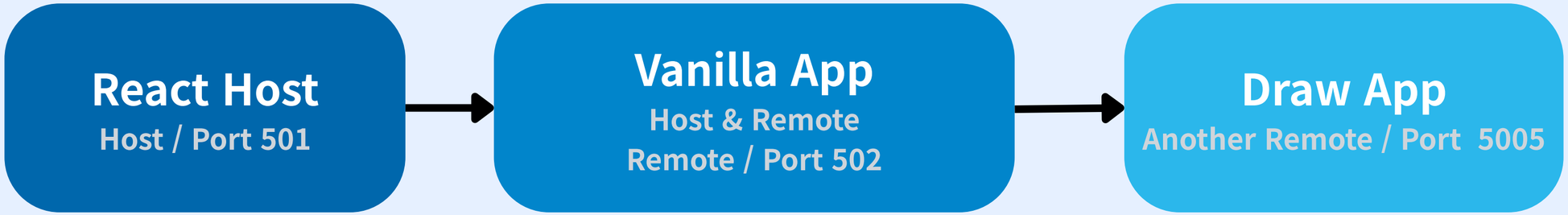
5002 port

5005 port

5.기술 시연 및 코드 설명 - MF 동작과정

Host이자 Remote

```
exposes: {
  './bootstrap': './js/bootstrap.js',
},
remotes: {
  remoteWidget: 'http://localhost:5004/assets/remoteEntry.js',
  remoteDraw: 'http://localhost:5005/assets/remoteEntry.js', //
},
```



출처 : Chat GPT 생성

6. iframe & MF - 공통점

독립적인 개발 및 배포

런타임 통합

기술 스택의 유연성

거대한 서비스를 작게 쪼개어
효율적으로 관리 및 배포

7. iframe VS Module Federation

격리성

완벽한 격리 vs 유기적 통합

통합 단위

문서 기반 vs 모듈 기반

통신 및 데이터 흐름

간접 통신 vs 직접 호출

리소스 효율성

중복 로드 vs 의존성 공유

UI/UX

UI 파편화 vs 자연스러운 처리

- 타 서비스 통합
- 안전한 격리
- 레거시 시스템의 통합
- 단순 기능 삽입

활용 분야

- 일관된 UI/UX
- 높은 성능
- 원활한 데이터 공유
- 협업 능률 향상

우리은행

상황에 맞는 전략적 도입이 중요

우리은행

감사합니다.

우리은행

Q&A