除了 Qiankun， 这些微前端框架你也应该知道

一、微前端现状

1.1 来源

微前端的概念是由 `ThoughtWorks` 在2016年提出的，它借鉴了微服务的架构理念，核心在于将一个庞大的前端应用拆分成多个独立灵活的小型应用，同时可以解决一些iframe的潜在问题。

1.2 意义

不可否认，微前端的设计起源于后端微服务，很多前端比较新的概念都来自于后端，比如Typescript，DDD领域驱动设计等等。

搞前端嘛，不折腾也不叫前端

微前端解决方案也给我们提供如下特性：

- 单个前端部分可独立开发、测试和部署；
- 无需重新构建即可添加、移除或替换单个前端部分；
- 不同的前端部分可使用不同的技术构建；
- 解决iframe硬隔离的一些问题

目前工作中遇到的项目有 `50%` 都已经应用上了该技术，这也体现了微前端带来的价值。

1.3 架构万能图

二、主流实现方案

2.1 qiankun为代表

- demo可参考：qiankun 使用方式代码Github

主应用

- 只有主应用需要安装qiankun，子应用不需要
- name，entry，container，activeRule，当判断页面路由匹配到activeRule时，就去动态创建script，把entry中的文件加载出来，因为子应用mount生命周期判断了渲染的#app，所以就可以把内容渲染到自定义的container中
- activeRule 则是和 window.location.pathname匹配，通过一级路由标识子应用

子应用接入

- 判断是否有传入的container，来判断是要渲染到主应用的#app，还是自己的#app
- 需要打包成一个umd格式的库：为了能通过window['app-name1']拿到子应用声明的生命周期，配合子应用的export的生命周期

实现原理

1、监视路由

- window.location.pathname等相关变化，触发接下来的匹配逻辑

2、匹配子应用

- 重写路由window.popState, replaceState，在保留原有功能的基础上，增加子应用entry映射相关逻辑
- 针对变化的路由，匹配子应用

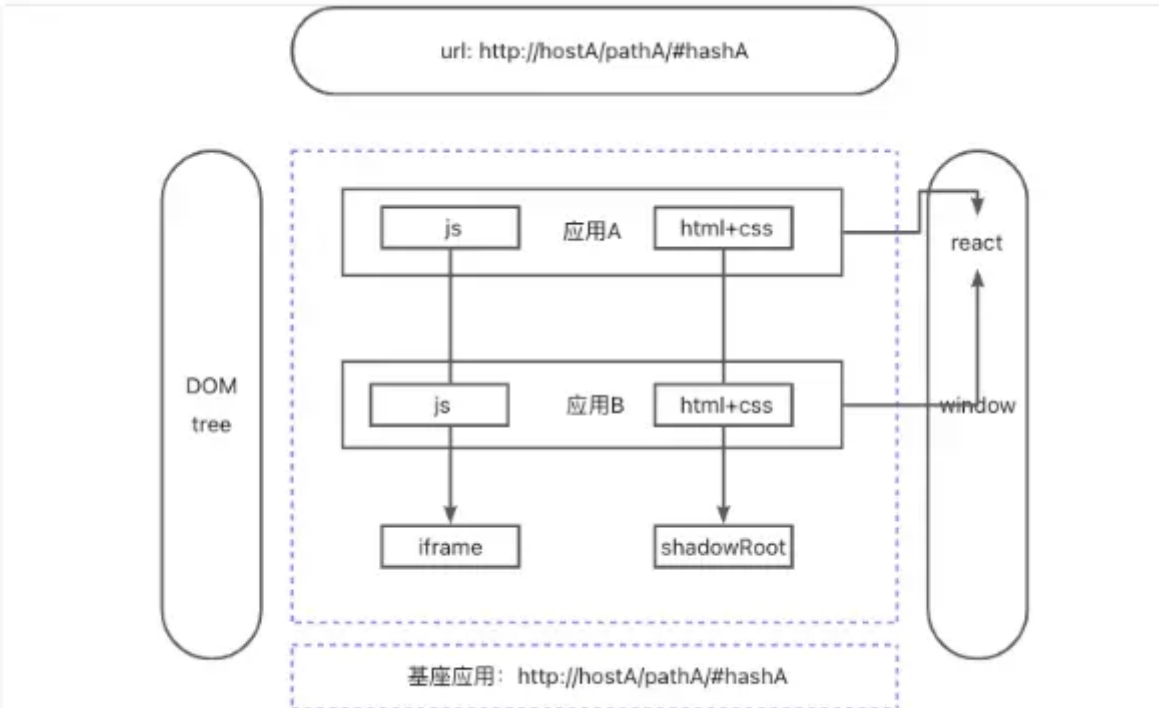
3、加载子应用

- import-html-entry 解析入口文件中的html 和 script
- 动态创建script去执行jsCode
- 通过umd模块获取子应用，调用子应用mount方法，render子应用

4、渲染子应用

- 把js和html，渲染到提前预留的#app容器中

2.1 以无界为代表



webComponent

- 浏览器原生API，实现方式这里copy一下小满的：了解webComponents
- 通过webComponent动态加载子应用，目前主流用同样方案的是无界和Micro App

```
window.onload = () => {
    class WuJie extends HTMLElement {
        constructor() {
```

```
4               super()
5               this.init()
6               this.getAttr('url')
7           }
8           init() {
9             const shadow =  this.attachShadow({ mode: "open" }) //开启影子dom 也就是样式隔离
10            const template = document.querySelector('#wu-jie') as HTMLTemplateElement
11            console.log(template);
12
13            shadow.appendChild(template.content.cloneNode(true))
14          }
15          getAttr (str:string) {
16             console.log('获取参数',this.getAttribute(str));
17
18          }
19
20          //生命周期自动触发有东西插入
21          connectedCallback () {
22             console.log('类似于vue 的mounted');
23          }
24          //生命周期卸载
25          disconnectedCallback () {
26                console.log('类似于vue 的destory');
27          }
28          //跟watch类似
29          attributeChangedCallback (name:any, oldVal:any, newVal:any) {
30              console.log('跟vue 的watch 类似 有属性发生变化自动触发');
31          }
32
33      }
34
35      window.customElements.define('wu-jie', WuJie)
36  }
```

沙箱

- 和qiankun sanbox沙箱不同的是，无界采用iframe做为沙箱方案，micro-app 0.x版本采用with沙箱，1.x支持vite后，支持和无界同样的ifame沙箱。
- 听说有大佬用v8封装了个沙箱？面试和一个大佬聊的，没太懂，感觉很牛逼

2.3 以webpack-module-federation为代表

- Emp

- Module Federation | webpack 中文文档
- vite-plugin-federation

`模块联邦` 主要是一种去中心化的思想，也可以用来做服务拆分， 实现原理比较复杂，主要涉及到以下几个方面：

1. **模块接口定义**

在需要共享的模块中，通过 module.exports 或 export 将需要共享的模块封装成一个模块接口，并将其在模块系统中注册。

2. **共享模块的描述信息**

在需要共享模块的应用程序中，通过使用 ModuleFederationPlugin 插件，将需要共享的模块的描述信息以 JSON 格式写入配置中。描述信息包括需要共享的模块名称、模块接口、提供共享模块的应用程序的 URL 等。

3. **共享模块的加载**

在需要使用共享模块的应用程序中，通过 webpack 的 container 远程加载共享模块的代码，并将其封装成一个容器。容器在当前应用程序中的作用是在容器中运行共享模块的代码，并按照描述信息将导出的模块接口暴露出来。容器本身是一个 JavaScript 运行时环境，它可以在需要使用共享模块的应用程序中被动或主动加载。

4. **远程模块的执行**

在容器中加载共享模块的代码后，容器需要将其执行，并将执行过程中产生的模块接口导出。为了实现这个目的，容器会利用 webpack 打包时在编译过程中生成的一个特殊的运行时代码，即 remoteEntry.js，通过 script 标签远程加载到当前应用程序中。在这个特殊的运行时代码中，会封装一些与容器通信的方法，例如 remote 方法，可以用于按需加载模块、获取模块接口等。

综上，webpack-module-federation 基于这些原理，实现了多个独立的应用程序之间的模块共享和远程加载，从而可以实现高度解耦、可扩展的架构。

三、各大框架

3.1 Single-Spa

作为比较早的微前端框架，single-spa只是实现了加载器、路由托管。沙箱隔离并没有实现。最早在第一家公司，项目就采用基于single-spa定制的一套自己框架，主要是也是通过proxySandBox实现沙箱隔离。

3.2 QianKun

QianKun 基于 single-spa ，阿里系开源的微前端框架，应该也是大家接触最多的了， `社区比较活跃` ，这点比较重要。

QianKun对 `single-spa` 方案进行完善，主要的完善点：

- 子应用资源由 js 列表修改进为一个 `url` ，大大减轻注册子应用的复杂度
- 实现应用隔离，完成 `js` 隔离方案 （ `window` 工厂） 和 `css` 隔离方案 （类 `vue` 的 `scoped` ）
- 增加资源预加载能力，预先子应用 `html` 、 `js` 、 `css` 资源缓存下来，加快子应用的打开速度

> - 2.x 不支持Vite，由于实现原理冲突，很难支持，参考Issue：想问一下，未来是否考虑支持 vite #1257
> - qiankun 3.0加入了支持，刚开始alpha阶段：github.com/umijs/qiank…

3.3 Mirco-App

Micro App 是京东出的一款基于 Web Component 原生组件进行渲染的微前端框架，不同于目前流行的开源框架，它从组件化的思维实现微前端，旨在降低上手难度、提升工作效率。

- 官方demo：Micro App
- 正式版本0.8版本， 1.0版本还是beta阶段，但是维护者在issue比较活跃
  - 0.x版本 vite支持不是很好，使用的时候需要关闭沙箱
  - 1.x版本 支持vite，需要采用iframe沙箱模式，这点和wujie的方案一样了，都是 webComponent + iframe
- MicroApp - 对vite支持的相关说明

接入

```
// 主 main.js

import microApp from "@micro-zoe/micro-app";

microApp.start();

// 加载子应用 a.js

import React from "react";

export default function AReact() {

return <micro-app name="ARact" url="http://127.0.0.1:4173/" iframe></micro-app>;
}

```

3.4 无界

- 官方demo：无界react-demo展示

无界是腾讯推出的一款微前端解决方式。它是一种基于 Web Components + iframe 的全新微前端方案，继承iframe的优点，补足 iframe 的缺点，让 iframe 焕发新生。

Web Components 是一个浏览器原生支持的组件封装技术，可以有效隔离元素之间的样式，iframe 可以给子应用提供一个原生隔离的运行环境，相比自行构造的沙箱 iframe 提供了独立的 window、document、history、location，可以更好的和外部解耦

接入

- 安装WujieReact即可，引入组件即可，和micro-app类似

```
<WujieReact
  width="100%"
  height="100%"
  name="xxx"
  url={xxx}
  sync={true}
  fetch={fetch}
  props={props}
  beforeLoad={beforeLoad}
  beforeMount={beforeMount}
  afterMount={afterMount}
  beforeUnmount={beforeUnmount}
  afterUnmount={afterUnmount}
></WujieReact>
```

3.5 Garfish

Garfish 字节跳动出品，当时出现的时候还是比较火热的。但是感觉有点半成品，没啥issue，没啥人用

3.6 EMP

- EMP是欢聚时代基于 Webpack5 Module Federation 搭建的微前端方案，更多的是去支持webpack，用的人也不多。
- 《react项目如何使用和接入EMP》 · efoxTeam/emp Wiki · GitHub

3.7 module-federation

- Module Federation | webpack 中文文档
- 需要加载打包后的产物，启动preview模式
- JS沙箱、样式隔离需要自己做
- 按照如下方式构建后，加载子应用，就和加载一个组件一毛一样

- 使用 `@originjs/vite-plugin-federation` 也能在vite架构下实现模块联邦

> 体验非常好，完全的spa切换体验，非常丝滑

remote

- 路由直接引入远程组件加载 `const AReact = lazy(() => import('remote_app_a/AReact'));`

```
import { defineConfig } from 'vite';
import react from '@vitejs/plugin-react-swc';
import federation from '@originjs/vite-plugin-federation';

export default defineConfig({
  plugins: [
    react(),
    federation({
      name: 'remote-app-a',
      filename: 'remoteEntry.js',
      // 需要暴露的模块
      exposes: {
        './AReact': './src/App.tsx',
      },
      shared: ['react'],
    }),
  ],
  build: {
    target: 'esnext',
  },
});


```

host

```
import { defineConfig } from 'vite';
import react from '@vitejs/plugin-react-swc';
import federation from '@originjs/vite-plugin-federation';

export default defineConfig({
  plugins: [
    react(),
    federation({
      name: 'host-app',
      filename: 'remoteEntry.js',
```

```
      remotes: {
        remote_app_a: "http://127.0.0.1:4174/assets/remoteEntry.js",
      },
      shared: ['react'],
    }),
  ],
  build: {
    target: 'esnext',
  },
});

```

3.8 其他

- ice-lab/**icestark** 飞冰的微前端方案，公司有个2年前的项目已经在用了
- teambit/**bit** 无中文文档，国外挺火

四、如何选择

- 自由度更高：module-federation
  - 需要自定义实现css隔离、js沙箱、路由劫持等功能
- 用的最多：qiankun
  - 相对比较成熟，社区活跃
  - webpack体系、接入相对比较重
- 接入更流畅：wujie、micro-app
- 基于react +vite技术栈，我们最终选择更新更活跃，文档更丰富的micro-app