如□□□□□□□

— 微前端的实现思路及应用

蜂泰科技 | 徐 健

□ □

□□□□□□

□□□□□□□□□□□□

□□

- □□
- □□□□□□
- □□□□□
- □□□□□□□□□□

- □□□□□□□
- □□□□□□□□□
- Q&A

> 脱离实际使用场景的技术改造没有太大价值并且很难走远。

- 技术债务：技术栈陈旧不堪，腐化速度越来越快；
- 耦合性高：多个业务团队同时维护一个巨石应用；
- 管理效率低：多个应用入口分散，权限分散；

- 开发、测试、部署、维护等环节繁琐，效率低；
- 改动成本高，出错率上升；
- 重复建设，造成资源浪费；
- 缺乏统一管理，使用成本上升；

□□□□□□

> 到了项目后期，怎么优化都不如进行项目拆分。

方案	描述	优点	缺点
Nginx路由转发	通过Nginx配置反向代理来实现不同路径映射到不同应用	简单、快速	在切换应用时会触发浏览器刷新，影响体验，配置繁复
iframe嵌套	每个子应用嵌套一个iframe	实现简单，子应用之间天然隔离互不影响	iframe的样式显示、兼容性等都具有局限性
Web Components	采用纯Web Components技术编写组件	每个子应用拥有独立的script和css，可单独部署	对于历史系统改造成本高，子应用通信较为复杂易踩坑
组合式应用路由分发	每个子应用独立构建和部署，运行时由父应用来进行路由管理，应用加载，启动，卸载，以及通信	体验良好，可无感知切换，子应用相互隔离	需要解决样式冲突、变量对象污染、通信机制等技术点

□□□□□□□

> 微前端是一种多个团队通过独立发布功能的方式来共同构建现代化 web 应用的技术手段及方法策略。-- Michael Geers

- 技术解耦

主框架不限制接入应用的技术栈，技术灵活；

- 独立开发、部署

微应用仓库独立，前后端可独立开发；

- 增量升级

复杂场景时，非常容易实施渐进式重构的手段和策略；

- 独立运行时

每个微应用之间状态隔离，运行时状态不共享；

□□□□□□□□□

> 微前端是一种类似微服务的架构，将微服务的理念应用到前端。

中心化路由-注册中心

注册中心：服务提供方要注册通告服务地址，服务的调用方要能发现目标服务。

这点和 后端的微服务类似。

我们需要在一个地方，统一维护路由和服务的对应关系。

分配一个路由给子应用：

```
// router.js
import Vue from 'vue'
import VueRouter from 'vue-router'
import MyPage from './my-page.vue'

Vue.use(VueRouter)

const routes = [
  {
    path: '/my-page/*',
    name: 'my-page',
    component: MyPage,
  },
]

export default routes
```

□□□□□□□□□

> 基座统一管理生命周期，触发相应的生命周期事件。

- 生命周期管理

微前端的中心服务称为：基座。

基座需要控制服务的加载，卸载。这时候对于子服务而言，就涉及到子服务的应用的生命周期。

一般需要包括以下功能：

- 资源加载、渲染
- 子应用挂载、卸载
- 全局监听、事件更新

- **qiankun** 框架的生命周期管理：

```
// main.js
/**
 * bootstrap只会在微应用初始化时调用一次。通常做一些全局变量初始化。
 */
export async function bootstrap() {}

/**
 * 应用每次进入都会调用 mount方法，通常会触发应用的渲染方法。
 */
export async function mount(props) {}

/**
 * 应用每次切出 /卸载会调用的方法，通常会卸载微应用的应用实例。
 */
export async function unmount(props) {}

/**
 * 可选生命周期钩子，仅使用 loadMicroApp 方式加载微应用时生效。
 */
export async function update(props) {}
```

□□□□□□□□□

JS沙箱和样式隔离，确保子应用样式、全局变量、事件等不干扰。

JS沙箱和样式隔离

JS沙箱通过Proxy代理子应用的全局对象，防止应用之间全局变量的冲突，记录或清空子应用的全局副作用函数，也可以向子应用注入全局变量用于定制化处理。

样式隔离是指对子应用的link和style元素的css内容进行格式化处理，确保子应用的样式只作用域自身，无法影响外部。

micro app 应用隔离:

```
▼<main class="ant-layout-content ant-pro-basicLayout-content ant-pro-basicLayout-has-header">
  ▼<div class="ant-row react16" style="row-gap: 0px;">flex
    ▶<div class="ant-col ant-col-6 btn-con">…</div>flex
    ▼<div class="ant-col ant-col-18 app-con-react16">flex
      ▶<h3>…</h3>
      ▼<micro-app name="react16" url="http://localhost:3001/" baseurl="/micro-app/demo/react16">
        ▶<micro-app-head>…</micro-app-head>
        ▶<micro-app-body>…</micro-app-body>
          <!--inline script extract by micro-app-->
          <!--inline script extract by micro-app-->
        </micro-app>
      </div>
    </div>
  </main>
```

基座应用

子应用

□□□□□□□

对比	single-spa	qiankun	Micro App
框架体积	20kb	94kb	30kb
渲染原理	基于路由监听	路由监听+手动渲染	CustomElement
数据通信机制	✗	基于props属性传递	基于发布订阅 +CustomEvent
接入成本	高	中	低
多框架兼容	√	√	√
JS沙箱	✗	√	√
样式隔离	✗	√	√
shadowDom	✗	√	√
预加载	✗	√	√
静态资源地址补全	✗	✗	√
元素隔离	✗	✗	√
插件系统	✗	✗	√

FAQ

□□□□ !

💪 □□□□□□□□□□

□□□□ □ □□

□□□□□□□□□□□□□https://sli.dev