无界微前端框架

无界微前端方案基于 WebComponent 容器 + iframe 沙箱

能够完善的解决适配成本、样式隔离、运行性能、页面白屏、子应用通信、子应用保活、多应用激活、vite 框架支持、应用共享等

一.主应用搭建

```
npx create-react-app base
```

```
import { BrowserRouter , Routes,Route,Link} from
'react-router-dom'
import Vue3 from './views/Vue3.js'
import React18 from './views/React18.js';
function App() {
  return (
    <div className="App">
        <BrowserRouter>
          <Link to='/vue'>Vue应用</Link>
          <Link to='/react'>React应用</Link>
          <Routes>
            <Route path='/vue' element=
{<Vue3/>}></Route>
```

```
            <Route path='/react' element=
{<React18/>}></Route>
          </Routes>
        </BrowserRouter>
    </div>
  );
}
```

二.子应用搭建

```
vue create m-vue
npx create-react-app m-react
```

```
import { useEffect, useRef } from "react";
import { startApp,destroyApp } from 'wujie'
export default function React18() {
    const myRef = useRef(null);
    let destroy = null;
    const startAppFunc = async () => {
        destroy = await startApp({
            url:'http://localhost:3001',
            name:'ReactApp',
            el: myRef.current
        });
    };
    useEffect(() => {
        startAppFunc()
```

```
            return ()=>{
                if(destroy){
                    destroyApp(destroy);
                }
            }
    })
    return <div style={{ width: "100%", height:
'100%' }} ref={myRef} />;
}
```

三.通用React组件

```
import { useEffect, useRef } from "react";
import { startApp, destroyApp } from 'wujie'
export default function WujieReact(props) {
    const myRef = useRef(null);
    let destroy = null;
    const startAppFunc = async () => {
        destroy = await startApp({
            ...props,
            el: myRef.current
        });
    };
    useEffect(() => {
        startAppFunc()
        return () => {
            if (destroy) {
                destroyApp(destroy);
            }
        }
    })
    const { width, height } = props;
    return <div style={{ width, height }} ref=
{myRef} />;
}
```

```
import WujieReact from
"../components/WujieReact.js";
export default function React18() {
    return <WujieReact name="ReactApp"
url="http://localhost:3001/"></WujieReact>
}
```

四.无界实现方案

> 定义WebComponet容器，后续用于承载HTML及CSS内容

1.实现简版WuJie

```
<!DOCTYPE html>
<html lang="en">


<head>
```

```
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible"
content="IE=edge">
    <meta name="viewport" content="width=device-
width, initial-scale=1.0">
    <title>Document</title>
</head>

<body>
    <div>基座的div</div>
    <div id="container"></div>
    <script>
        const strTmplWithCss = `
            <!DOCTYPE html>
            <html lang="zh-CN">
            <head>
                <meta charset="UTF-8">
                <meta http-equiv="X-UA-
Compatible" content="IE=edge">
                <meta name="viewport"
content="width=device-width, initial-scale=1.0">
                <title>Document</title>
            </head>
            <body>
                <div id="inner">你好，世界</div>
                <style>
                    div{
                        color:red
                    }
                </style>
            </body>
            </html>
        `;
        const strScript = `
            window.a = 100;
            console.log('子应用',window.a); // 子应用获取window属性
            const ele = document.querySelector('#inner'); // 查询子应用的dom元素
            console.log(ele);
        `;
        // 创建 iframe, 返回 iframe 对象
        function createIframe() {
            const iframe = document.createElement("iframe");
            iframe.src = 'about:blank'
            document.body.appendChild(iframe);
            return iframe;
        }
        function createSandbox() {
            const sandbox = {
                iframe: createIframe(),
                shadowRoot: null
            };
            return sandbox;
        }
        function injectTemplate(sandbox, 
template) {
            const div = 
document.createElement('div');
            div.innerHTML = template;
            sandbox.shadowRoot.appendChild(div);
        }

        // 创建脚本标签，执行脚本
        function runScriptInSandbox(sandbox, 
script) {
            const iframeWindow = 
sandbox.iframe.contentWindow;
            const scriptElement = 
iframeWindow.document.createElement("script");
            const head = 
iframeWindow.document.querySelector("head");

 Object.defineProperty(iframeWindow.Document.pro
totype, 'querySelector', {
                get: ()  =>  {
                    return new 
Proxy(sandbox.shadowRoot['querySelector'], {
```

```
                        apply(target, thisArg,
args) {
                            return
thisArg.querySelector.apply(sandbox.shadowRoot,
args);
                        }
                    })
                },
            });
            scriptElement.textContent = script;
            head.appendChild(scriptElement);
        }
        function createCustomElement() {
            class WujieApp extends HTMLElement {
                connectedCallback() {
                    // 1.创建 iframe 沙箱
                    const sandbox =
createSandbox();
                    // 2.将沙箱的 shadowRoot 作为
该元素的 shadowRoot
                    const shadowRoot =
this.attachShadow({ mode: "open" });
                    sandbox.shadowRoot =
shadowRoot;
                    // 将模板放入到沙箱中
                    injectTemplate(sandbox,
strTmplWithCss);
```

```
                    // 运行脚本
                    runScriptInSandbox(sandbox,
strScript);
                }
            }

 window.customElements?.define("wujie-app",
WujieApp);
            const contentElement =
document.createElement("wujie-app");

 container.appendChild(contentElement);
        }
        // 调用函数
        createCustomElement();
    </script>
</body>
</html>
```

2.defineWujieWebComponent

```
export function defineWujieWebComponent() {
  const customElements = window.customElements;
  // 如果没有定义过 wujie-app 则创建对应的组件
  if (customElements &&
!customElements?.get("wujie-app")) {
    class WujieApp extends HTMLElement {
      // 1.组件加载完成执行此方法
      connectedCallback(): void {
        if (this.shadowRoot) return; // 有
shadowDOM 直接return
        // 1)创建shadowDOM
        const shadowRoot = this.attachShadow({
mode: "open" });
        // 2)获取组件对应的实例（沙箱）
        const sandbox =
getWujieById(this.getAttribute(WUJIE_APP_ID));
        // 3)给shadowRoot定义BaseURI属性和
ownerDocument属性
        patchElementEffect(shadowRoot,
sandbox.iframe.contentWindow);
        // 4)在实例（沙箱）上保存shadowRoot
        sandbox.shadowRoot = shadowRoot;
      }
      // 2.组件卸载完成执行此方法
      disconnectedCallback(): void {
        // 调用沙箱的卸载逻辑
        const sandbox =
getWujieById(this.getAttribute(WUJIE_APP_ID));
        sandbox?.unmount();
      }
    }
    // 定义wujie-app组件
    customElements?.define("wujie-app",
WujieApp);
  }
}
```

3.startApp方法

- 创建Wujie实例 （iframe沙箱）
- 获取模板（模板渲染到shadowRoot）、样式（使用css-loader处理样式）、及脚本（使用js-loader处理，并在沙箱中执行js脚本）
- css、js的资源加载依旧通过fetch来实现

```
/**
 * 运行无界app
 */
export async function startApp(startOptions:
startOptions): Promise<Function | void> {
  // 根据name获取缓存的沙箱
  const sandbox =
getWujieById(startOptions.name);
  // setupApp设置的属性
  const cacheOptions =
getOptionsById(startOptions.name);
  // 合并缓存配置
  const options = mergeOptions(startOptions,
cacheOptions);
```

```
  const {
    name, // 应用名称
    url, // 应用路径
    html,
    replace,
    fetch,
    props, // 应用props
    attrs,
    degradeAttrs,
    fiber, // 是否开启fiber模式
    alive, // 保活
    degrade, // 是否降级
    sync,
    prefix, // css前缀
    el, // 挂载的元素
    loading,
    plugins,
    lifecycles, // 生命周期
  } = options;
  // 已经初始化过的应用，快速渲染......
  // 设置loading
  addLoading(el, loading);
  // 1.创建iframe沙箱
  const newSandbox = new WuJie({ name, url,
attrs, degradeAttrs, fiber, degrade, plugins,
lifecycles });
  // 调用前执行生命周期中beforeLoad方法
  newSandbox.lifecycles?.beforeLoad?.
(newSandbox.iframe.contentWindow);
  // 2.fetch url 拿到模板，额外的脚本及样式表（出了模
板之外的）
  const { template, getExternalScripts,
getExternalStyleSheets } = await importHTML({
    url,
    html,
    opts: {
      fetch: fetch || window.fetch,
      plugins: newSandbox.plugins,
      loadError:
newSandbox.lifecycles.loadError,
      fiber,
    },
  });
  // 处理html和样式
  const processedHtml = await
processCssLoader(newSandbox, template,
getExternalStyleSheets);
  // 激活沙箱
  await newSandbox.active({ url, sync, prefix,
template: processedHtml, el, props, alive,
fetch, replace });
  // 沙箱中运行js
  await newSandbox.start(getExternalScripts);
  return newSandbox.destroy;
```

```
}
```

4.初始化WuJie

- `new WuJie`核心流程，创建一个iframe域名采用主域名，停止iframe加载

```
// 子应用：创建目标地址的解析，a标签、应用主机路径、应用路由路径
const { urlElement, appHostPath, appRoutePath }
= appRouteParse(url);
// 主应用路径
const { mainHostPath } = this.inject;
// 创建iframe
this.iframe = iframeGenerator(this, attrs,
mainHostPath, appHostPath, appRoutePath);

// 降级方案，生成iframe的 window、document、
location对象
if (this.degrade) {
  const { proxyDocument, proxyLocation } =
localGenerator(this.iframe, urlElement,
mainHostPath, appHostPath);
  this.proxyDocument = proxyDocument;
  this.proxyLocation = proxyLocation;
} else {
   // 1.proxyWindow:生成代理window，处理this问题
```

```
    // 2.proxyDocument:iframe中的创建方法采用主应用的，查找等方法采用的是shadowRoot中的
    // 3.proxyLocation:代理location属性
   const { proxyWindow, proxyDocument, proxyLocation } = proxyGenerator(
     this.iframe,
     urlElement,
     mainHostPath,
     appHostPath
   );
   this.proxy = proxyWindow;
   this.proxyDocument = proxyDocument;
   this.proxyLocation = proxyLocation;
}
this.provide.location = this.proxyLocation;
// 将实例（沙箱）添加到wujie中
addSandboxCacheWithWujie(this.id, this);
```

```
export function iframeGenerator(
  sandbox: WuJie,
  attrs: { [key: string]: any },
  mainHostPath: string,
  appHostPath: string,
  appRoutePath: string
): HTMLIFrameElement {
  // 创建iframe元素
  const iframe =
window.document.createElement("iframe");
  // 设置iframe的属性(iframe采用的是主域名), 并添加到
主应用
  const attrsMerge = { src: mainHostPath, style:
"display: none", ...attrs, name: sandbox.id,
[WUJIE_DATA_FLAG]: "" };
  setAttrsToElement(iframe, attrsMerge);
  window.document.body.appendChild(iframe);

  const iframeWindow = iframe.contentWindow;
  // 给iframe注入变量 (iframeWindow.__WUJIE =
wujie;)
  patchIframeVariable(iframeWindow, sandbox,
appHostPath);
  // 停止iframe加载 (创建一个空的iframe)
  sandbox.iframeReady =
stopIframeLoading(iframeWindow).then(() => {
    initIframeDom(iframeWindow, sandbox,
mainHostPath, appHostPath);
    /**
     * 如果有同步优先同步, 非同步从url读取
     */
  });
  return iframe; // 返回iframe
}
```

```
function initIframeDom(iframeWindow: Window,
wujie: WuJie, mainHostPath: string, appHostPath:
string): void {
  // ....
  // 初始化base标签
  initBase(iframeWindow, wujie.url);
  // iframe history处理, 将history.pushState 和
history.replaceState方法重写，用于修改父应用的
history
  patchIframeHistory(iframeWindow, appHostPath,
mainHostPath);
  // 重写iframe中的事件监听, 例如history中常见的api
hashChange和popState
  patchIframeEvents(iframeWindow);
  if (wujie.degrade)
recordEventListeners(iframeWindow);
  // 监听iframe中url的变化，同步到window中
  syncIframeUrlToWindow(iframeWindow);
  // 对有些事件绑定给window，有些事件绑定给主应用中
  patchWindowEffect(iframeWindow);
  // 对document中的事件绑定也做处理
  patchDocumentEffect(iframeWindow);
  // 对节点的baseURI进行修复  例如插入和追加
  patchNodeEffect(iframeWindow);
  // 劫持相对路径转换成绝对路径
  patchRelativeUrlEffect(iframeWindow);
}
```

5.激活沙箱active

```
// 找到iframe的body
const iframeBody = rawDocumentQuerySelector.call(iframeWindow.document, "body") as HTMLElement;
// 创建webComponent丢到iframe中（会生成shadowRoot）
this.el = renderElementToContainer(createWujieWebComponent(this.id), el ?? iframeBody);
// 处理模板中的样式转化为style，移动到shadowRoot中。做到css隔离
await renderTemplateToShadowRoot(this.shadowRoot, iframeWindow, this.template);
```

6.开始执行脚本start

```
syncScriptResultList.concat(deferScriptResultLis
t).forEach((scriptResult) => {
  // 插入script标签到iframe中, （生成script标签，放入
到head中）
  scriptResult.contentPromise.then((content) =>
                                  this.fiber
                                  ?
requestIdleCallback(() => insertScriptToIframe({
...scriptResult, content }, iframeWindow))
                                  :
insertScriptToIframe({ ...scriptResult, content
}, iframeWindow)
                                 )

});
```

###