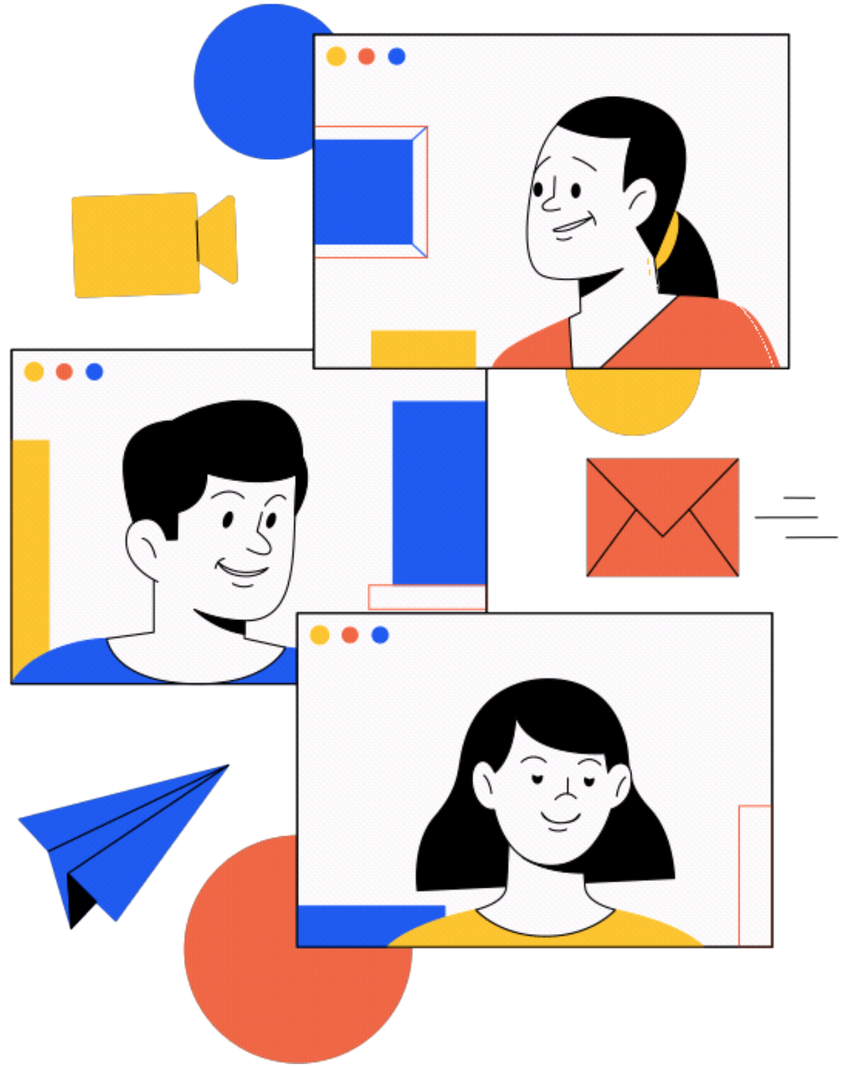
SE400.P11.PMCL

Báo cáo Cuối kì

ÁP DỤNG KIẾN TRÚC MICROFRONTEND VÀO ỨNG DỤNG ĐẶT PHÒNG KHÁCH SẠN BẰNG WEBPACK MODULE FEDERATION



GVHD: ThS. Đinh Nguyễn Anh Dũng

SVTH:

- Trần Tuấn Kiệt - 21522266
- Trần Thị Tuyết Mai - 21520340

Nội dung

1 Giới thiệu chung

2 So sánh với kiến trúc truyền thống

3 Các phương pháp tích hợp

4 Các loại tích hợp chính

5 Webpack Module Federation

6 Quy trình phát triển

7 Yêu cầu dẫn đến lựa chọn kiến trúc

8 Ứng dụng Microfrontend trong lĩnh vực đặt phòng khách sạn

9 Kết luận

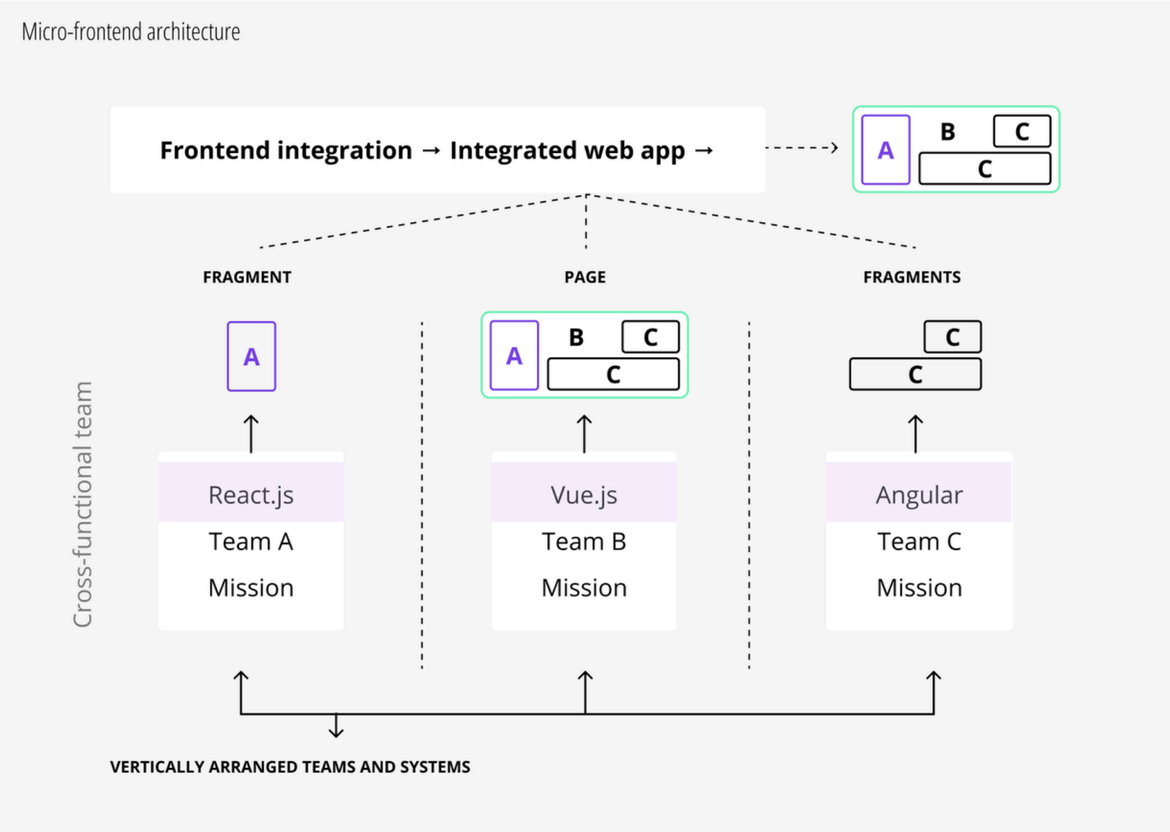
1 Giới thiệu chung

- Trong thời đại công nghệ số hiện nay, nhu cầu sử dụng các ứng dụng đặt phòng khách sạn ngày càng gia tăng, kéo theo sự cạnh tranh khốc liệt trong ngành du lịch và khách sạn.
- Việc phát triển các ứng dụng này không chỉ đơn thuần là cung cấp dịch vụ đặt phòng, mà còn phải đảm bảo trải nghiệm người dùng tốt nhất, tốc độ tải trang nhanh và **khả năng mở rộng linh hoạt**.
- Trong bối cảnh đó, kiến trúc **Microfrontend** đã nổi lên như một giải pháp tối ưu giúp các developer giải quyết những thách thức này.

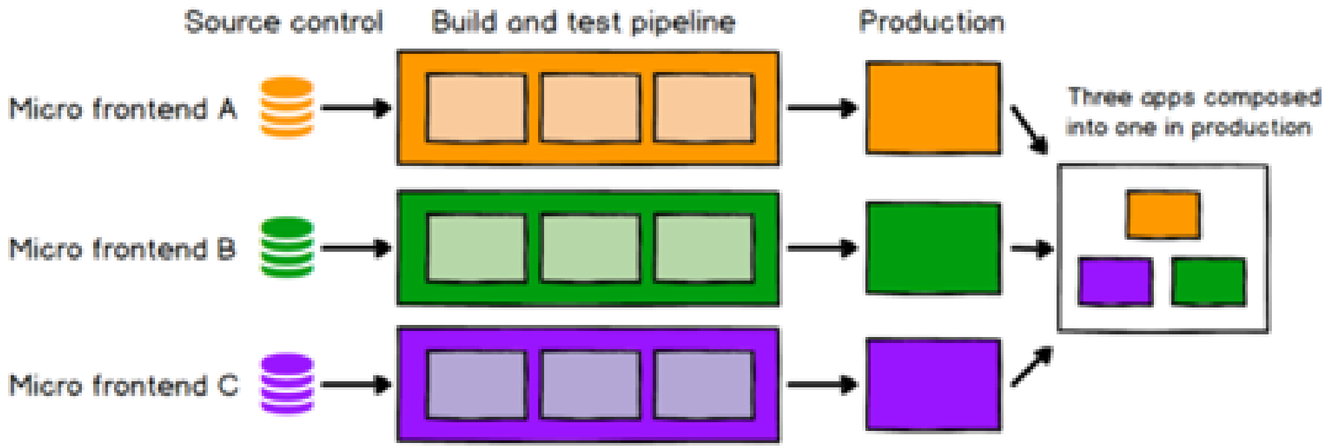
1 Giới thiệu chung

- **Microfrontend** là một phong cách **kiến trúc của phát triển web frontend**, trong đó **một ứng dụng được chia thành các tính năng** và **được phân phối độc lập**.
- **Mỗi microfrontend thường đại diện cho một tính năng hoặc một phần của giao diện người dùng**, cho phép các nhóm phát triển làm việc song song mà không bị ảnh hưởng lẫn nhau.
- Mỗi microfrontend có thể có kho source code, dependencies, automation tests và pipelines riêng.
- Điều này được thực hiện để cải thiện chất lượng phân phối và hiệu quả của các nhóm chịu trách nhiệm về code frontend.

1 Giới thiệu chung

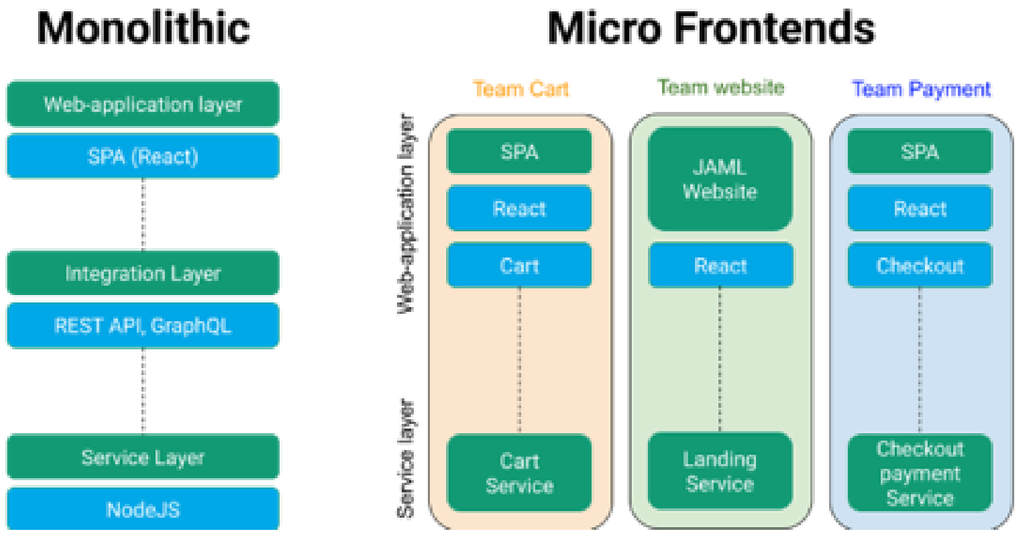


1 Giới thiệu chung

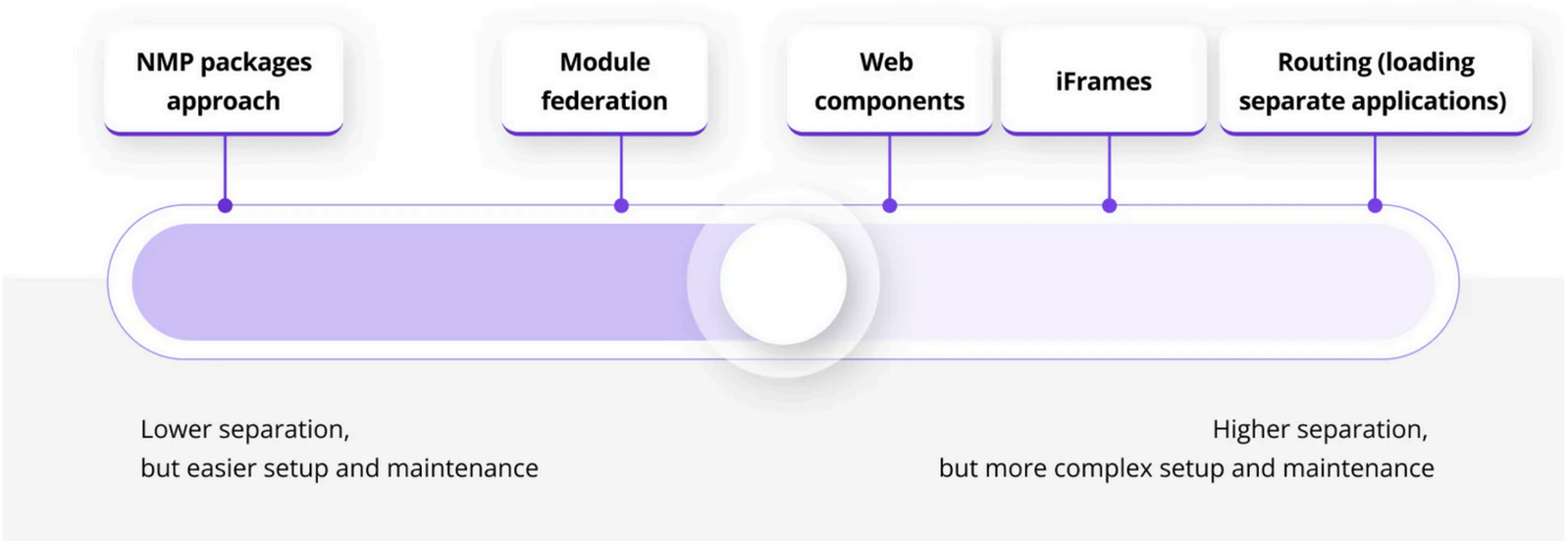


2 So sánh với kiến trúc truyền thống

- **Kiến trúc truyền thống**: Thường sử dụng một ứng dụng đơn nhất, trong đó tất cả các phần của ứng dụng được phát triển và triển khai cùng nhau. Điều này có thể dẫn đến việc khó khăn trong việc bảo trì, cập nhật và mở rộng ứng dụng.
- **Microfrontend**: Mỗi phần của ứng dụng được phát triển độc lập và có thể sử dụng công nghệ khác nhau. Điều này cho phép các nhóm phát triển tự do chọn công nghệ phù hợp nhất cho từng tính năng mà không làm ảnh hưởng đến các phần khác của ứng dụng.

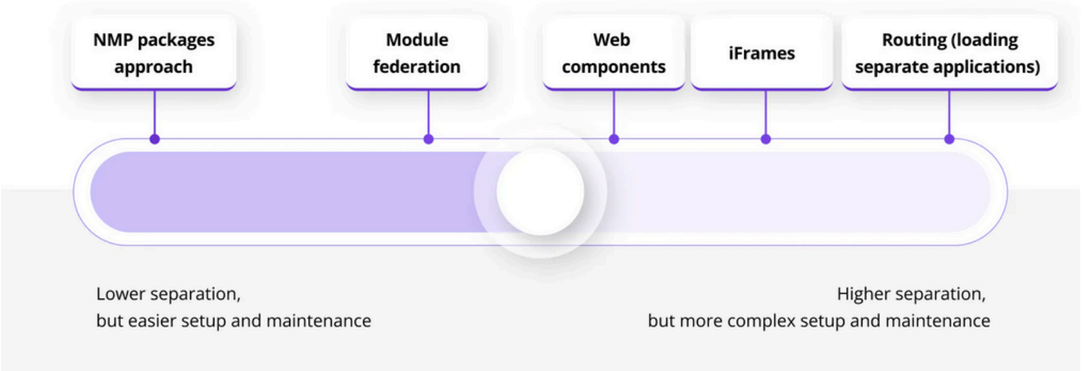


3 Các phương pháp tích hợp



3 Các phương pháp tích hợp

NPM

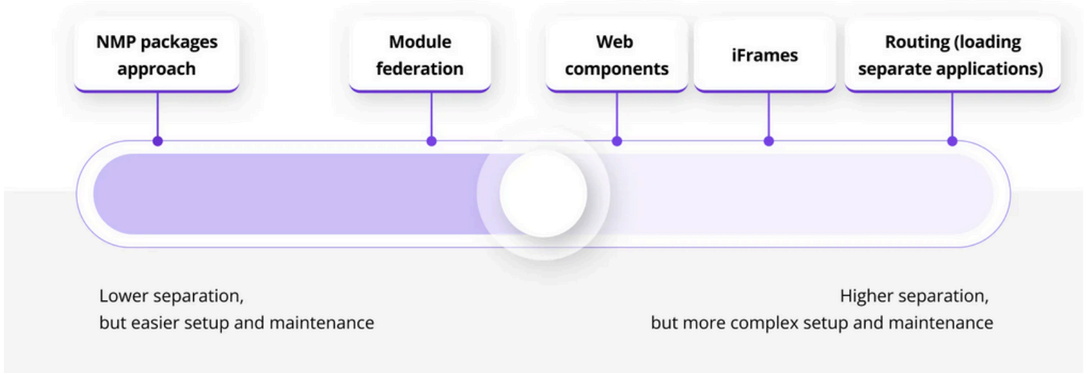


- Mỗi microfrontend được phát hành dưới dạng gói NPM và tích hợp vào shell*. Phương pháp này đơn giản nhưng gây phụ thuộc phiên bản và cần cập nhật shell khi release, làm chậm trễ các bản cập nhật.

* Kiến trúc shell application là phương pháp xây dựng các ứng dụng web tiến bộ hiện nay, tận dụng nhiều công nghệ khác nhau.

3 Các phương pháp tích hợp

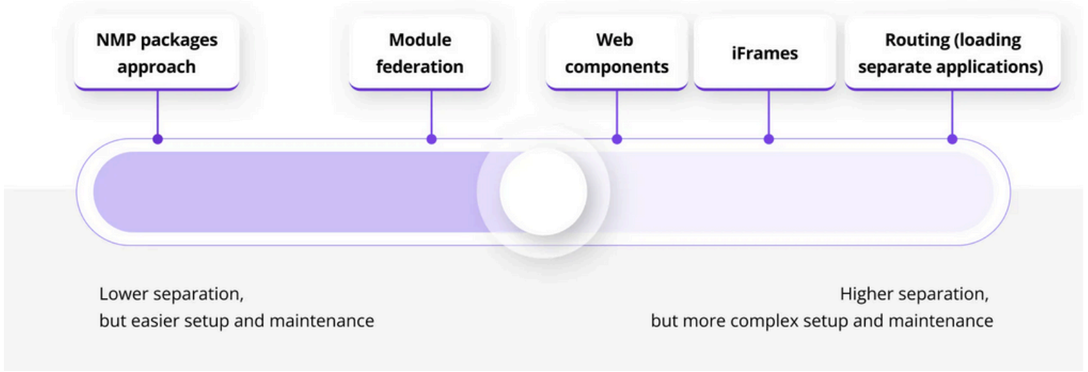
Module Federation



- Module Federation cho phép tải microfrontend khi chạy vào ứng dụng shell mà không phụ thuộc vào thời gian build.

3 Các phương pháp tích hợp

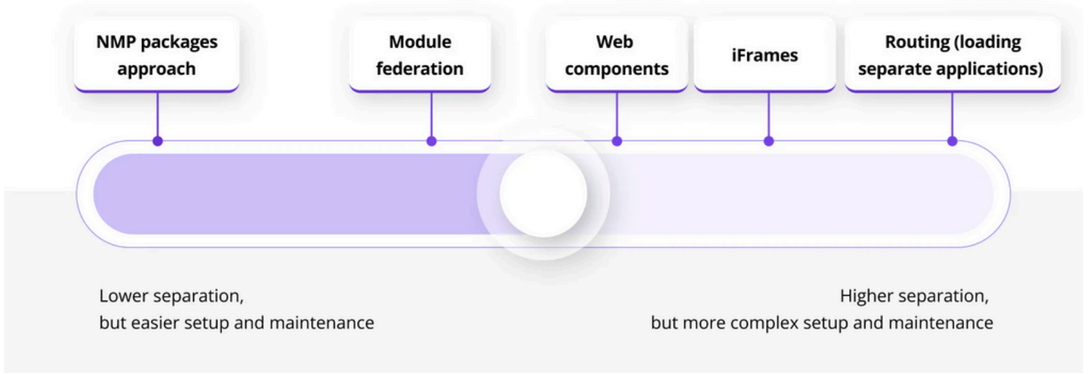
Web Components



- Các Web Components ngụ ý xây dựng từng microfrontend như một thành phần riêng biệt có thể được triển khai độc lập dưới dạng tệp .js. Ứng dụng load và hiển thị chúng trong các placeholders được tạo riêng trong bố cục của trang web.

3 Các phương pháp tích hợp

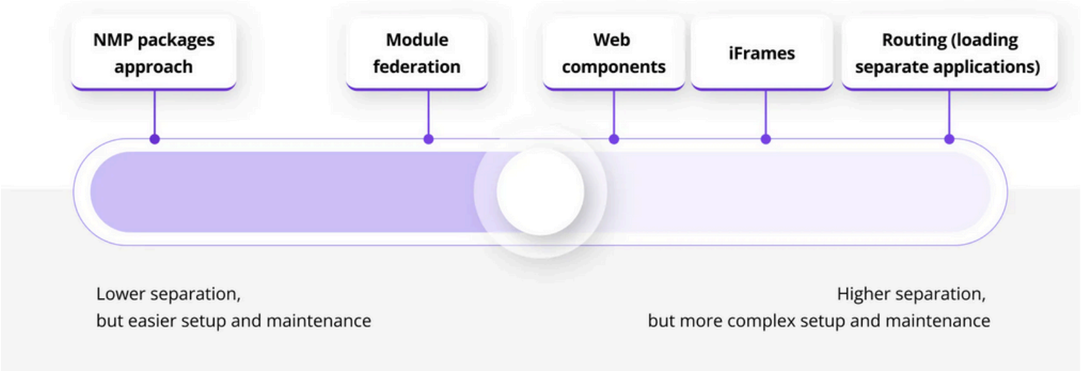
iFrames



- Phương pháp iframe nhúng từng microfrontend vào iframe của riêng nó và một ứng dụng trang đơn (SPA) hoạt động như một container. Về cơ bản, đây là một tài liệu HTML được đặt bên trong một tài liệu HTML khác.

3 Các phương pháp tích hợp

Routing

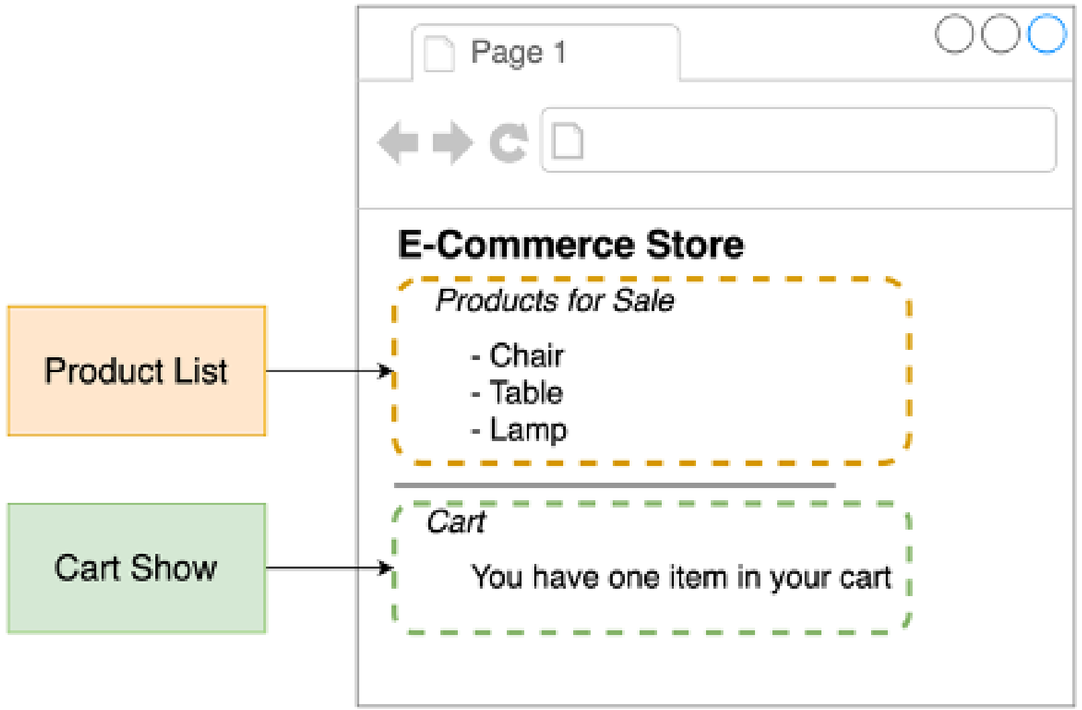


- Cách tiếp cận này tách ứng dụng dựa trên định tuyến, mỗi tuyến tải một ứng dụng độc lập qua API gateway hoặc CDN. Các microfrontend hoạt động như ứng dụng một trang, định tuyến bằng liên kết HTML hoặc shared shell application. Dữ liệu chia sẻ bị hạn chế và trang có thể tải lại hoặc không, tùy vào cách triển khai. Shared shell hoặc meta-framework như single-spa giúp hiển thị trang nhanh chóng mà không cần tải lại toàn bộ trang.

3 Các phương pháp tích hợp

Tình huống

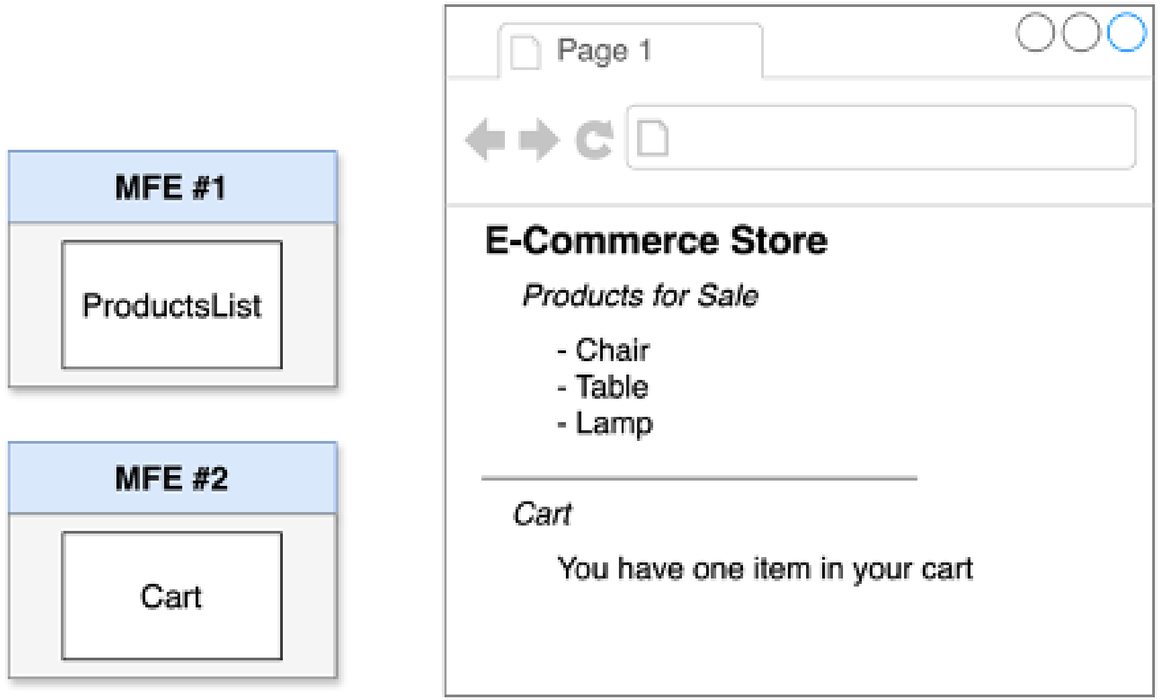
Giả sử chúng ta có một trang thương mai điện tử với 2 thành phần là danh sách sản phẩm và giỏ hàng



3 Các phương pháp tích hợp

Tình huống

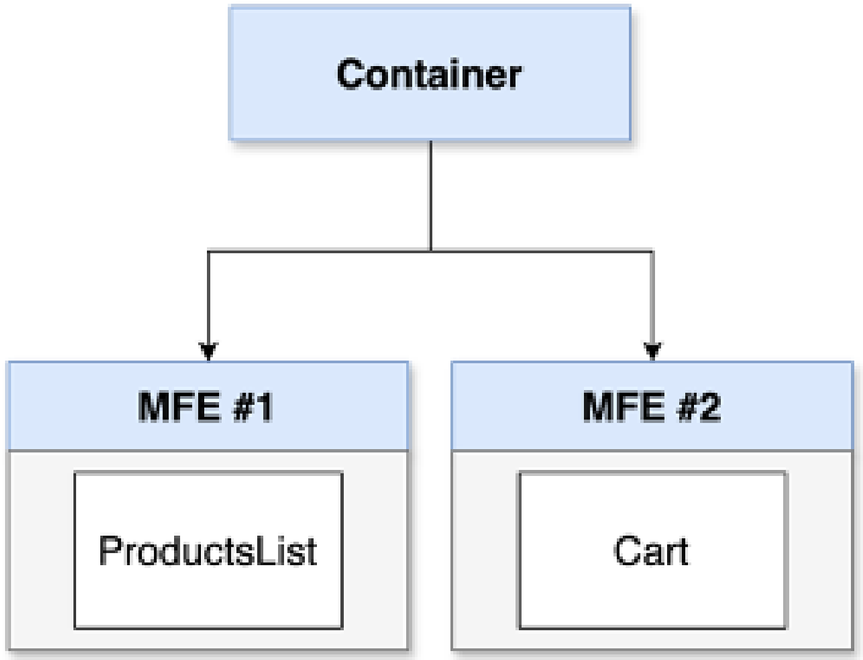
Ta sẽ chia ra được 2 microfrontend cho 2 thành phần này



3 Các phương pháp tích hợp

Tình huống

Nhìn một cách tổng quan, nó sẽ được biểu diễn như sau

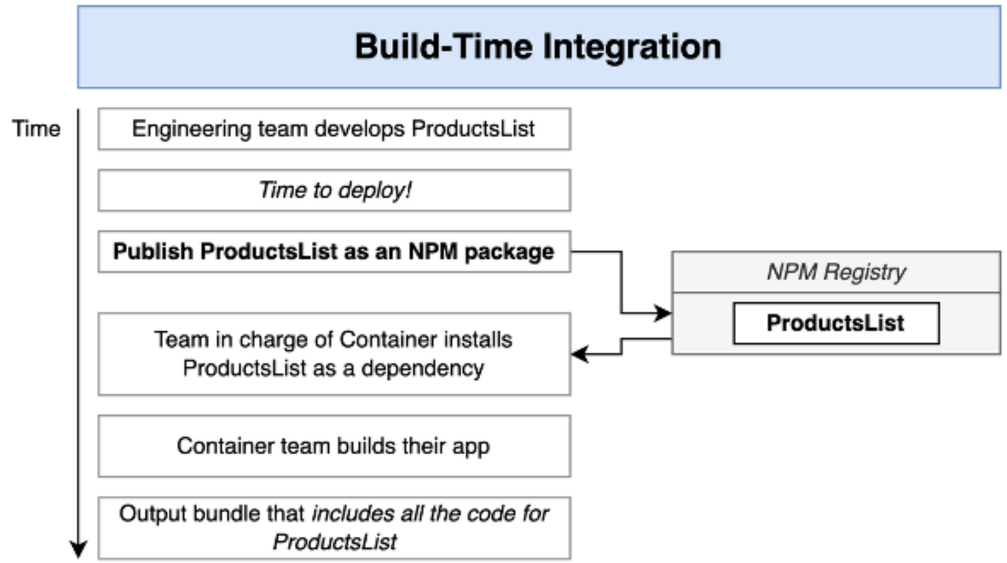


Nhưng vấn đề đặt ra là: **Container có thể truy cập vào mã nguồn trong MFE #1 và #2 khi nào và bằng cách nào?**

4 Các loại tích hợp chính

Build-time integration

- Trước khi Container được tải trong trình duyệt, nó sẽ được truy cập vào mã nguồn ProductsList

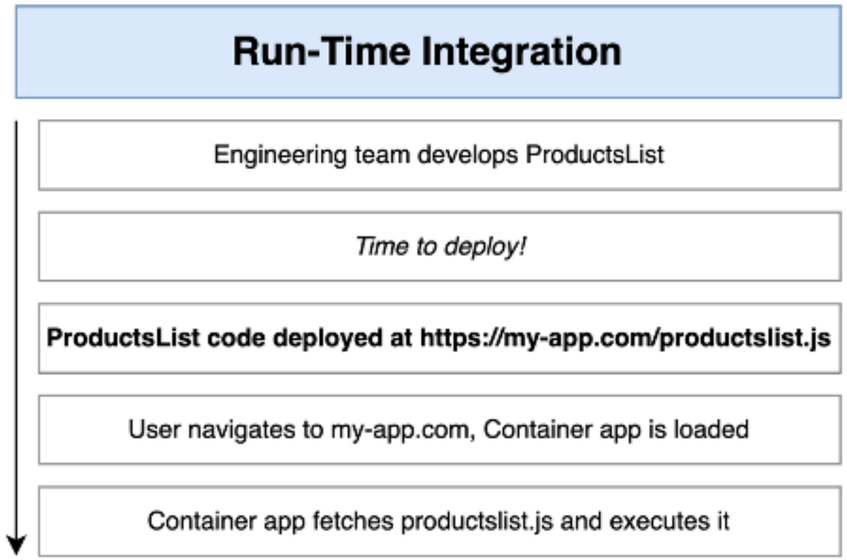


- Với cách tiếp cận này, sẽ có lợi ích là dễ cài đặt và dễ hiểu, nhưng Container phải được triển khai lại mỗi khi ProductsList được cập nhật và có ý định (cám dỗ) kết hợp chặt chẽ Container + ProductsList với nhau

4 Các loại tích hợp chính

Run-time integration

- Sau khi Container được tải trong trình duyệt, nó sẽ được truy cập vào mã nguồn ProductsList



- Với cách tiếp cận này, ProductList có thể được độc lập bất cứ lúc nào và có thể triển khai nhiều phiên bản khác nhau của ProductsList và Container có thể quyết định sử dụng phiên bản nào, nhưng công cụ và thiết lập phức tạp hơn nhiều.

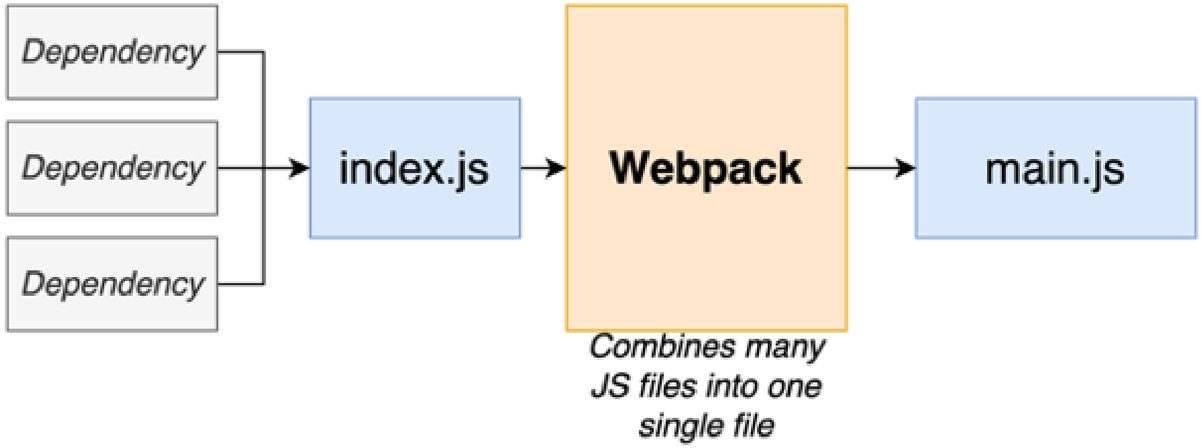
4 Các loại tích hợp chính

Server integration

- Trong khi gửi JS xuống để tải Container, máy chủ quyết định có bao gồm nguồn ProductsList hay không.

5 Webpack Module Federation

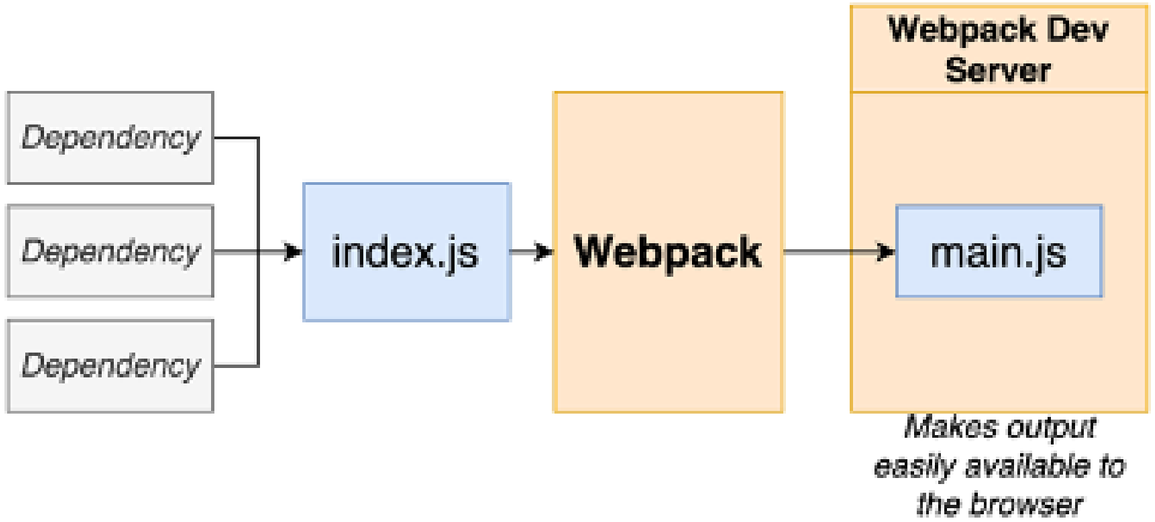
- Webpack là một trình đóng gói mô-đun mã nguồn mở và miễn phí cho JavaScript.
- Nó được tạo chủ yếu cho JavaScript, nhưng nó có thể chuyển đổi các asset giao diện người dùng như HTML, CSS và hình ảnh nếu các trình tải tương ứng được bao gồm.
- Webpack lấy các mô-đun có phụ thuộc và tạo các asset tĩnh biểu diễn các mô-đun đó.
- Node.js là bắt buộc để sử dụng Webpack.



5 Webpack Module Federation

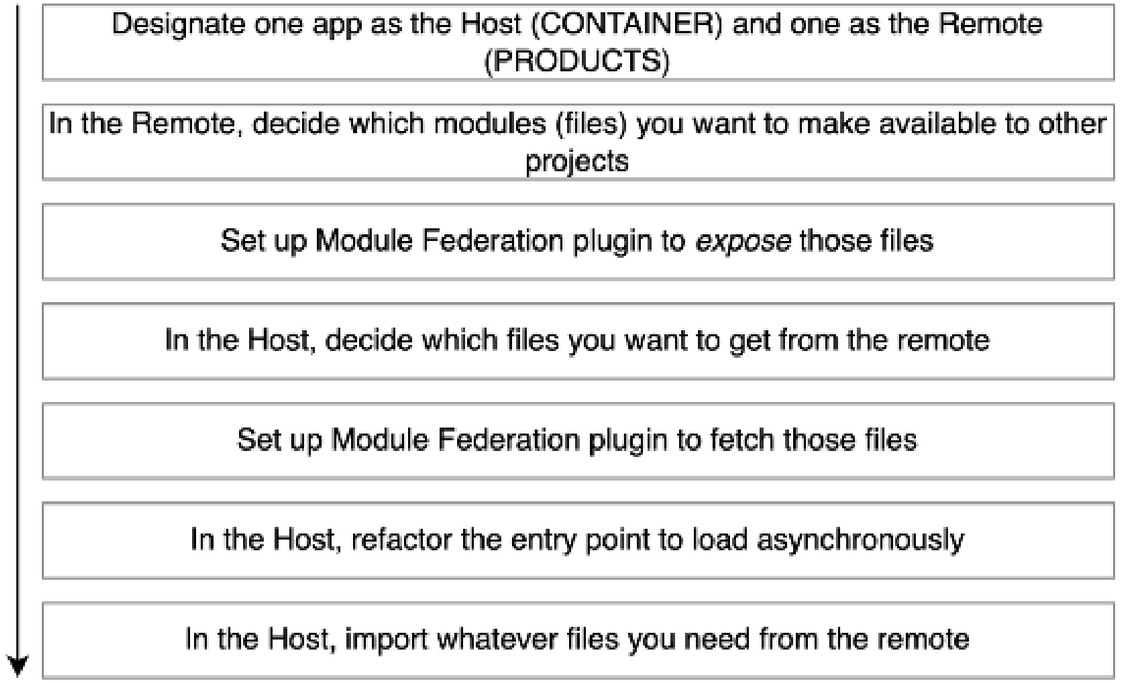
Webpack development server

- Webpack cũng cung cấp một máy chủ phát triển tích hợp, **webpack-dev-server**, có thể được sử dụng như một máy chủ HTTP để phục vụ các tệp trong khi phát triển.
- Nó cũng cung cấp khả năng sử dụng **hot module replacement (HMR)**, cập nhật mã trên trang web mà không yêu cầu nhà phát triển tải lại trang.



5 Webpack Module Federation

Các bước để cài đặt

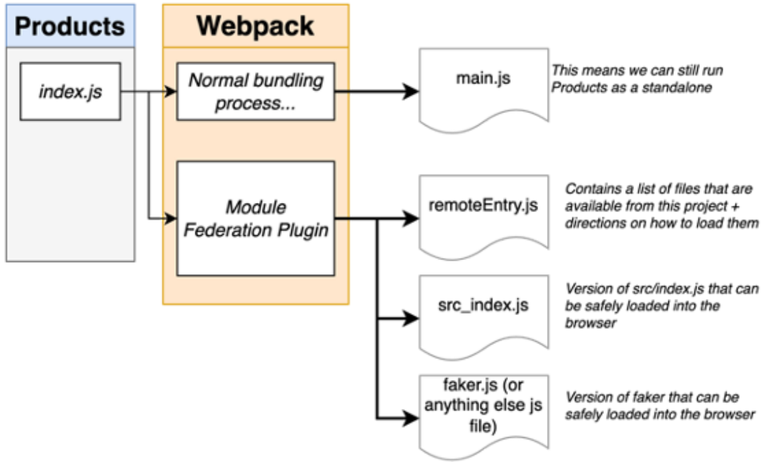


5 Webpack Module Federation

Quy trình đóng gói (bundling) của Webpack khi sử dụng Module Federation cho ứng dụng "Products" (Remote):

Kết quả sau khi chạy Webpack:

- main.js: File chính chứa mã đã được đóng gói của ứng dụng.
- remoteEntry.js: File chứa metadata và các thông tin cần thiết để các ứng dụng khác có thể tải các module từ "Products". Đây là file quan trọng nhất trong cơ chế chia sẻ module.
- src_index.js, faker.js (hoặc bất kỳ file JavaScript nào khác): Các file này đại diện cho các phần mã khác mà "Products" có thể sử dụng hoặc chia sẻ. Tất cả sẽ được quản lý bởi Webpack

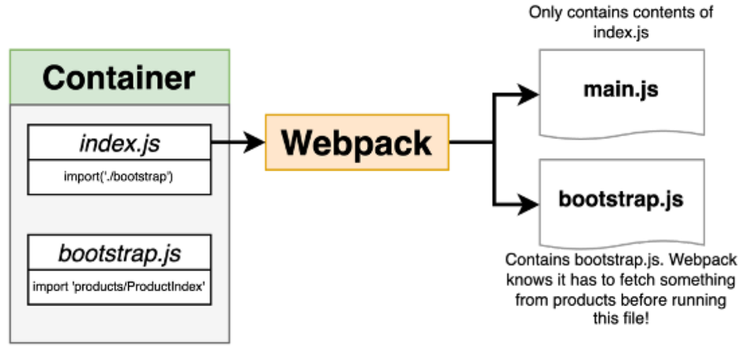


5 Webpack Module Federation

Quá trình sử dụng Webpack để gộp và đóng gói các tệp JavaScript từ các thành phần khác nhau

Webpack: Công cụ này được sử dụng để đóng gói (bundle) các tệp JavaScript từ Container thành các tệp độc lập:

- main.js
- bootstrap.js

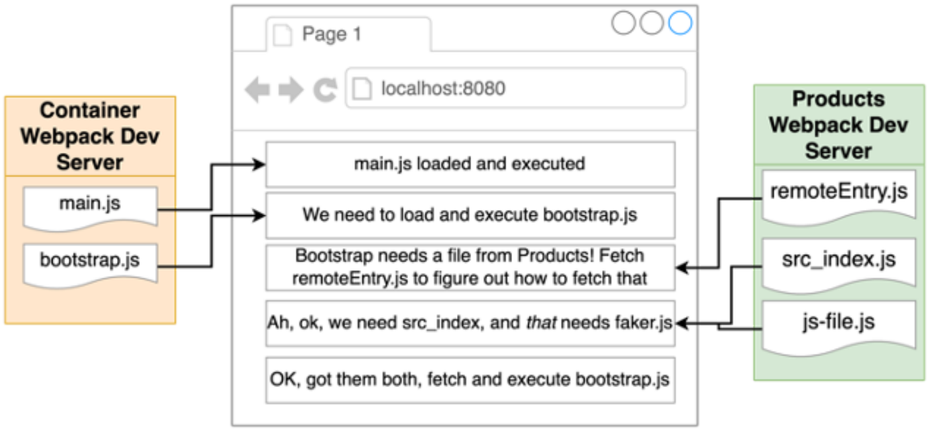


5 Webpack Module Federation

Quá trình tải và thực thi các tệp JavaScript giữa hai Webpack Dev Server khác nhau

Quá trình này diễn ra như sau:

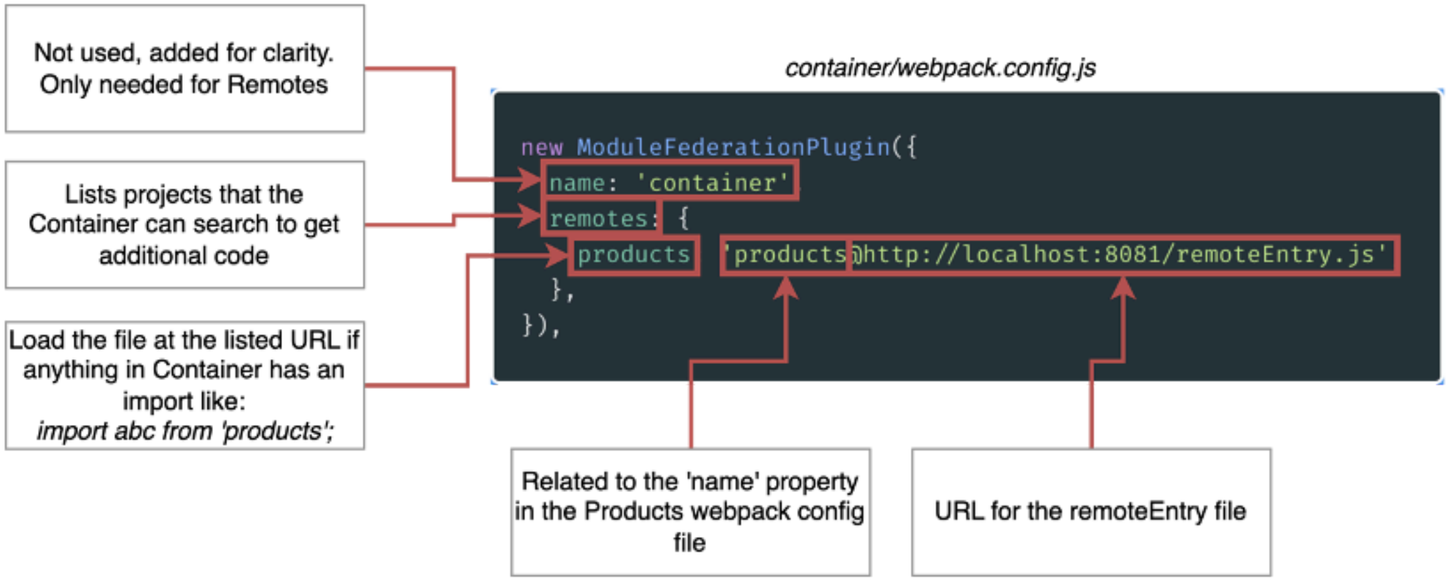
- Trình duyệt truy cập trang web tại localhost:8080.
- main.js được tải và thực thi trước.
- Sau đó, main.js yêu cầu tải và thực thi bootstrap.js.
- Tuy nhiên, bootstrap.js cần một tệp từ Products. Do đó, nó gửi yêu cầu tải remoteEntry.js từ Products Webpack Dev Server để biết cách tải các tệp khác.
- Sau khi remoteEntry.js được tải, nó cho biết rằng cần tải src_index.js, và src_index.js cũng yêu cầu thêm faker.js.
- Cuối cùng, khi tất cả các tệp cần thiết đã được tải, bootstrap.js được thực thi.



5 Webpack Module Federation

Hiểu hơn về các tuỳ chọn cấu hình - Đối với host

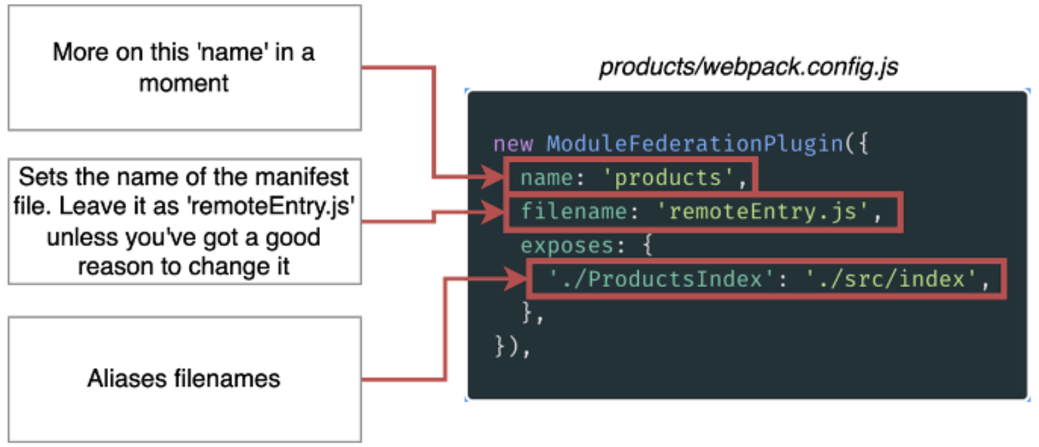
- Không cần phải quá quan tâm về name.
- remotes: Danh sách dự án mà Host có thể tìm để lấy mã.
- remotes.products: tải tệp ở URL được liệt kê nếu mọi thứ trong Host có một import kiểu import abc from ‘products’. Cái tên products này chính là name của Remote.



5 Webpack Module Federation

Hiểu hơn về các tuỳ chọn cấu hình - Đối với remote

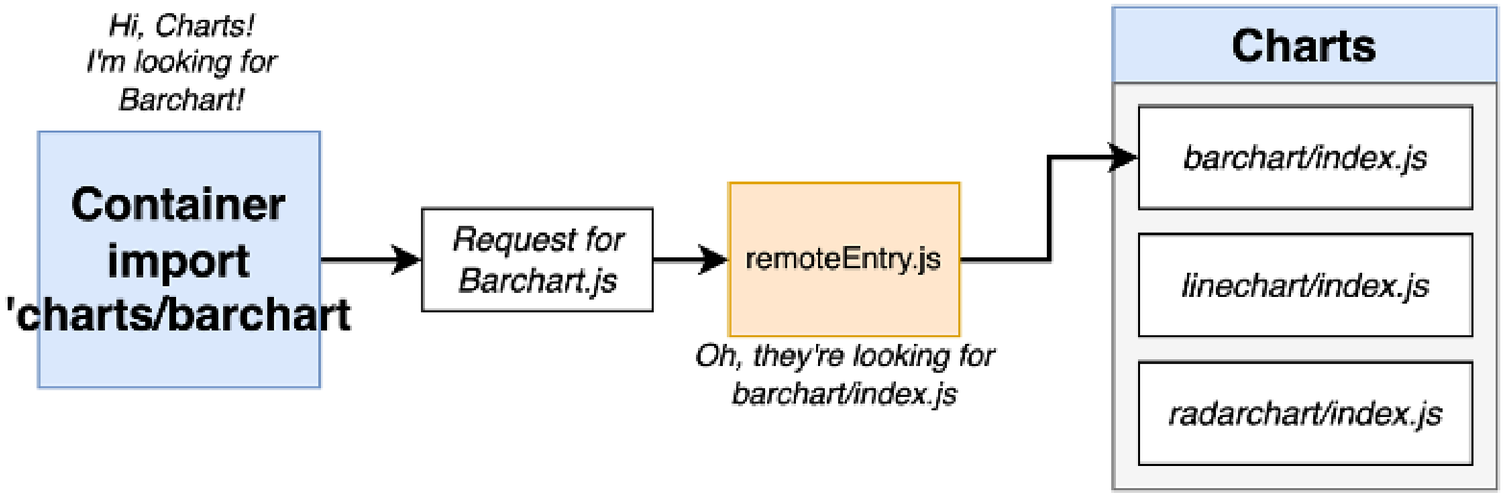
- name: đây là một thuộc tính vô cùng quan trọng vì nó quyết định tên remote mà cần được liệt kê ở Host.
- filename: đây là tên file manifest. Mặc định nên để là remoteEntry.js. Đây sẽ là file mà host sẽ cần phải liệt kê và tìm tới.
- exposes: liệt kê danh sách các thành phần cần được lấy ra trong filename. Và đặt cho chúng những alias phù hợp



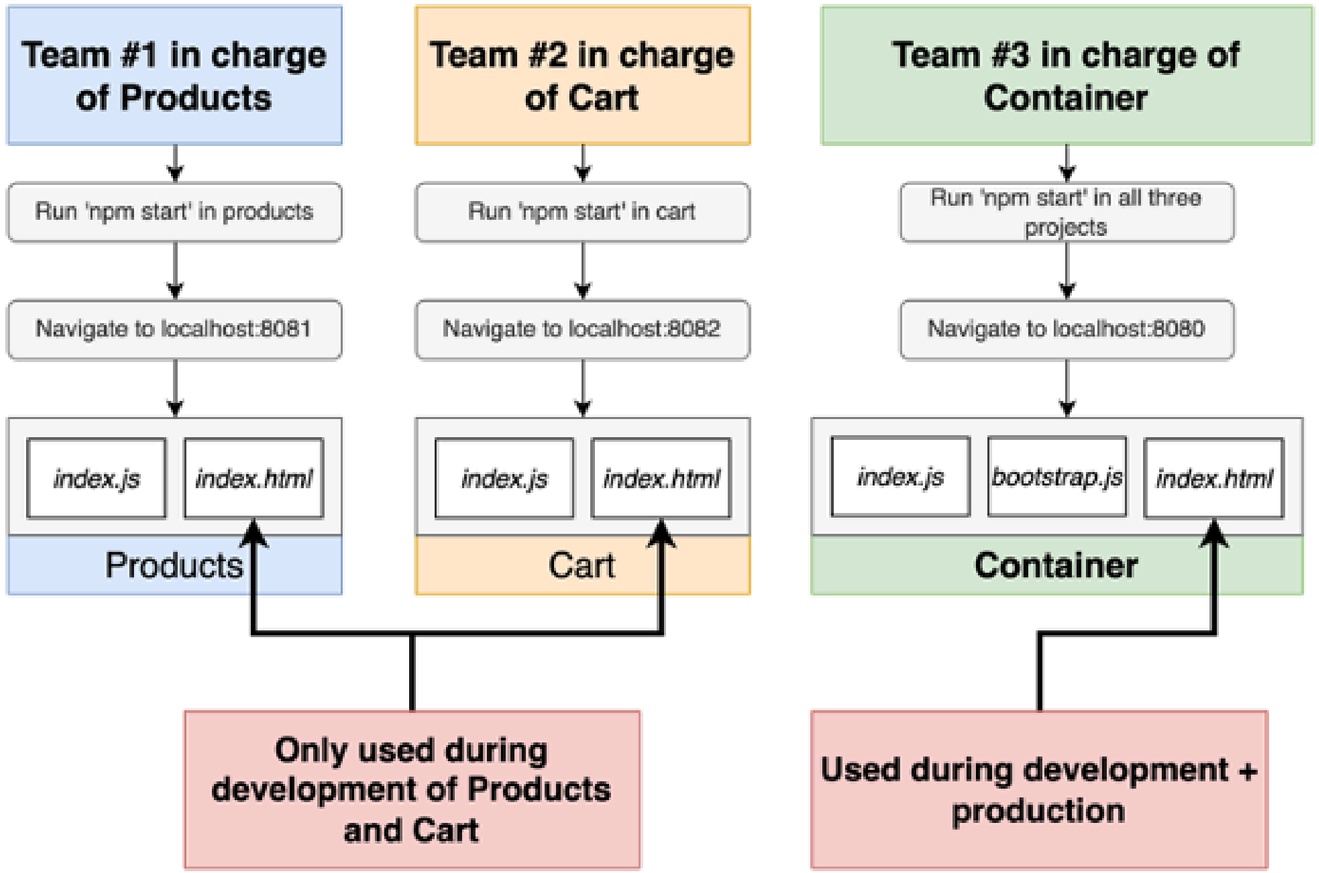
5 Webpack Module Federation

Tại sao nên có sự phân biệt giữa các alias?

- Tạo ra sự phân biệt đó để Host có thể tìm kiếm đúng và chính xác thành phần cần sử dụng tử Remote

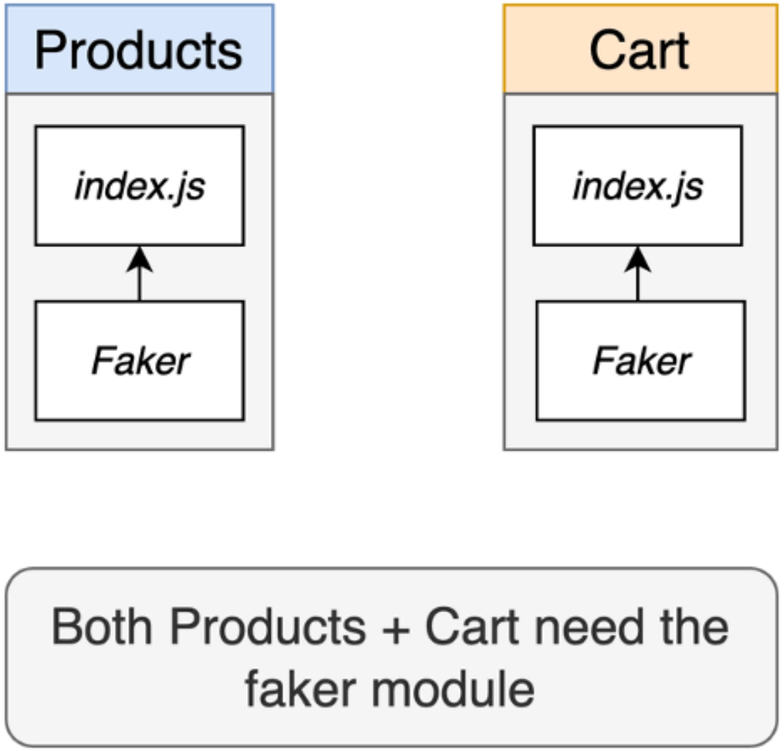


6 Quy trình phát triển



6 Quy trình phát triển

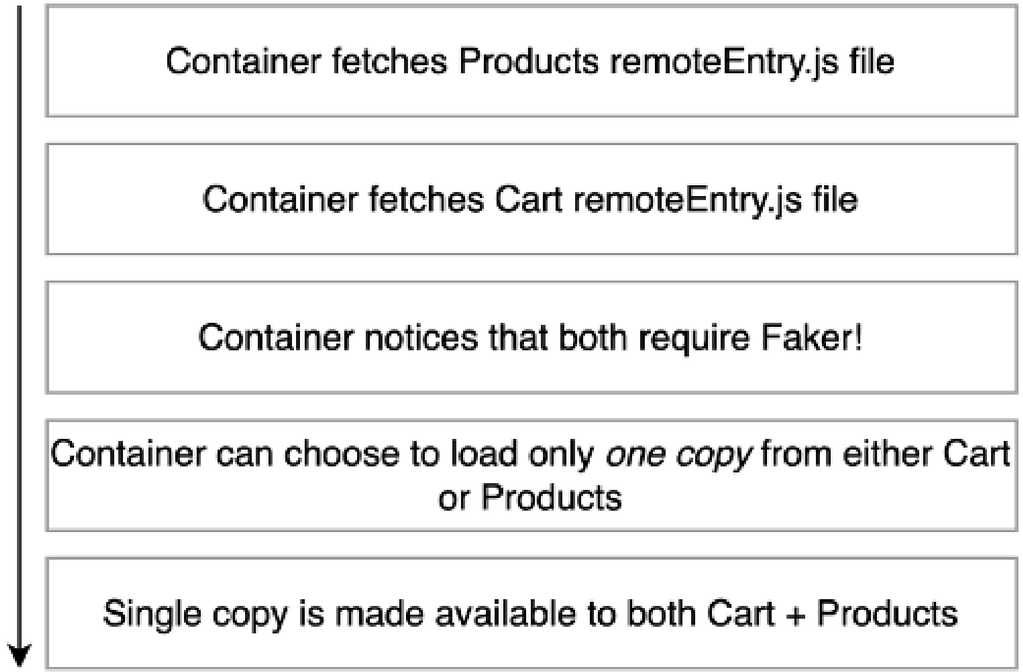
Chia sẻ các phụ thuộc giữa các ứng dụng (sub-project)



6 Quy trình phát triển

Chia sẻ các phụ thuộc giữa các ứng dụng (sub-project)

- Quy trình chia sẻ module sẽ được diễn ra như sau:

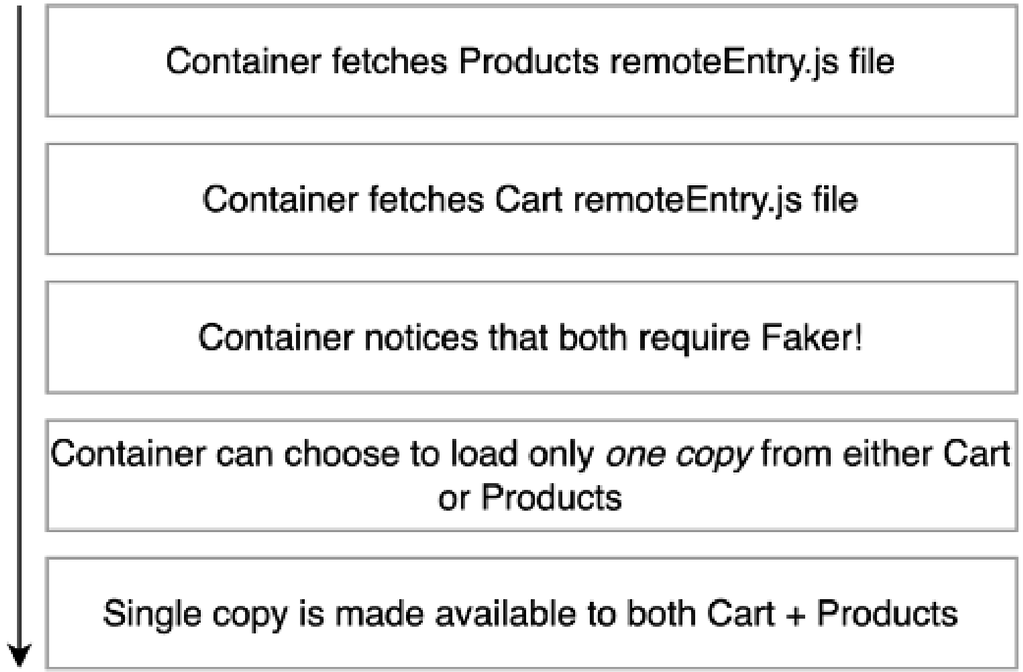


Vậy nếu các phụ thuộc tuy giống nhau về tên, nhưng có sự phân biệt về **version** thì sao?

6 Quy trình phát triển

Chia sẻ các phụ thuộc giữa các ứng dụng (sub-project)

- Quy trình chia sẻ module sẽ được diễn ra như sau:

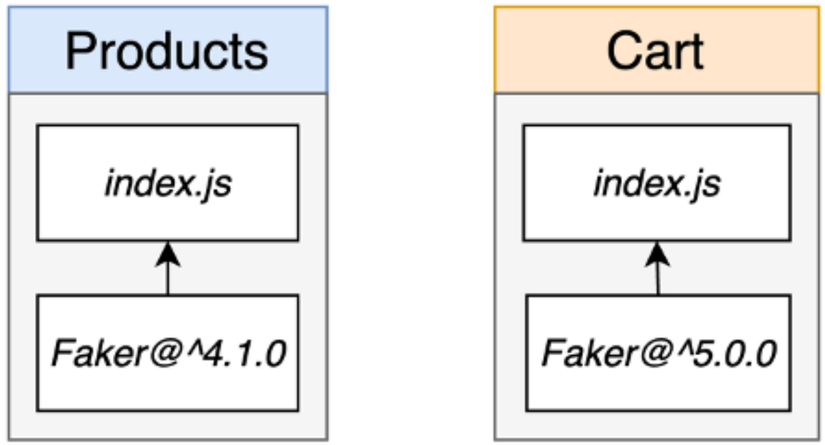


Vậy nếu các phụ thuộc tuy giống nhau về tên, nhưng có sự phân biệt về **version** thì sao?

6 Quy trình phát triển

Chia sẻ các phụ thuộc giữa các ứng dụng (sub-project)

- Các phiên bản phụ thuộc ở các Remote khác nhau

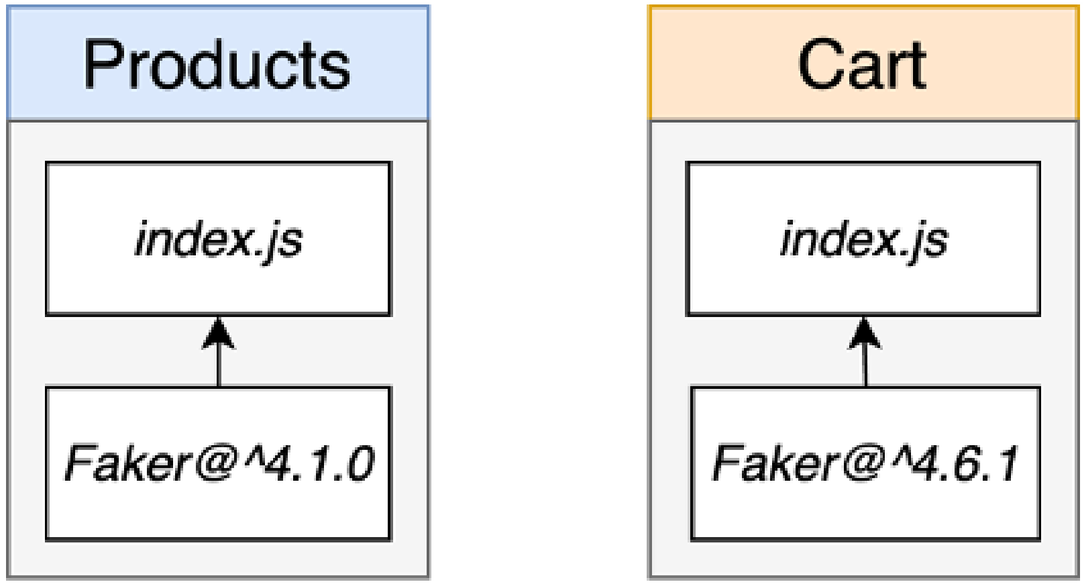


Trong trường hợp này, Webpack sẽ tự động load cả 2 module của 2 phụ thuộc có 2 phiên bản khác nhau này

6 Quy trình phát triển

Chia sẻ các phụ thuộc giữa các ứng dụng (sub-project)

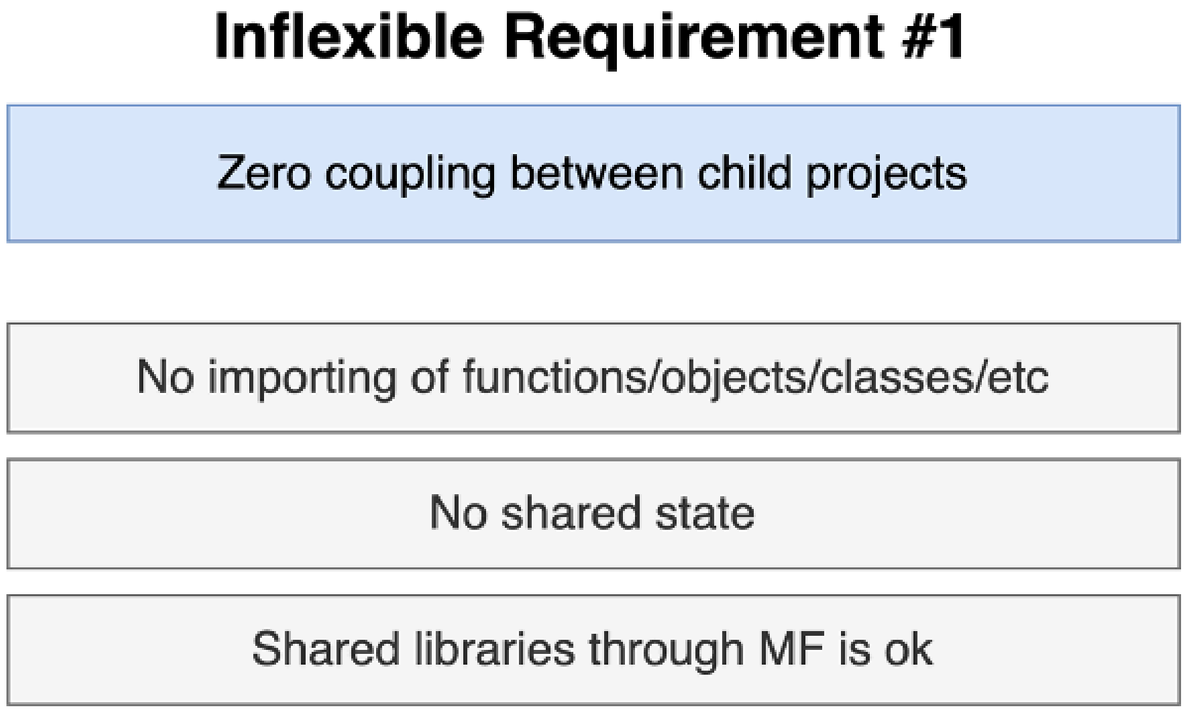
- Các phiên bản phụ thuộc ở các Remote khác nhau về phiên bản nhỏ

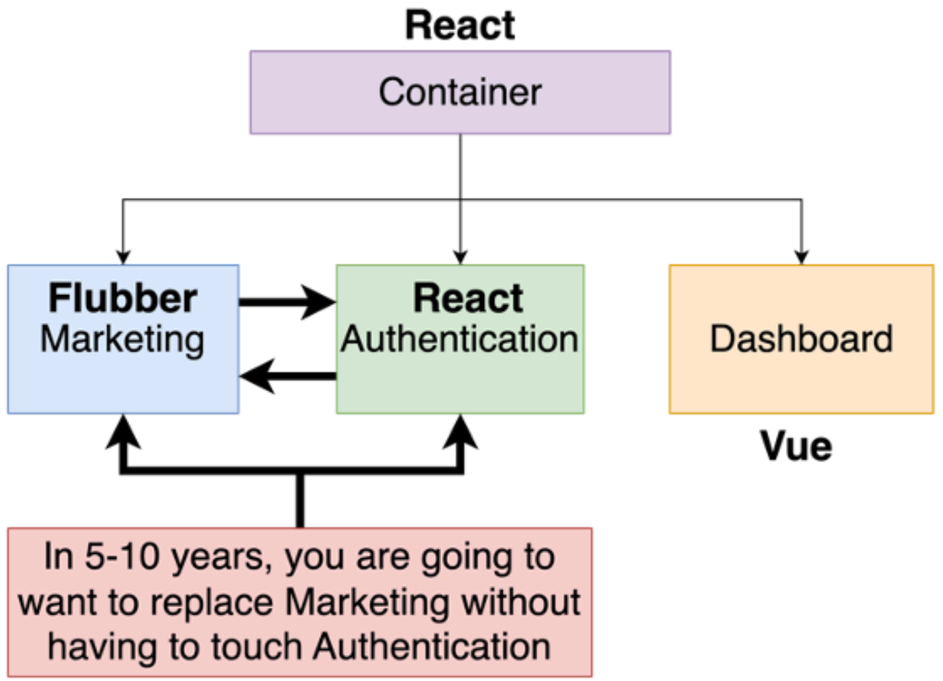


Trong trường hợp này, Webpack vẫn sẽ load một bản copy của bản phụ thuộc đó duy nhất.

7 Yêu cầu dẫn đến lựa chọn kiến trúc

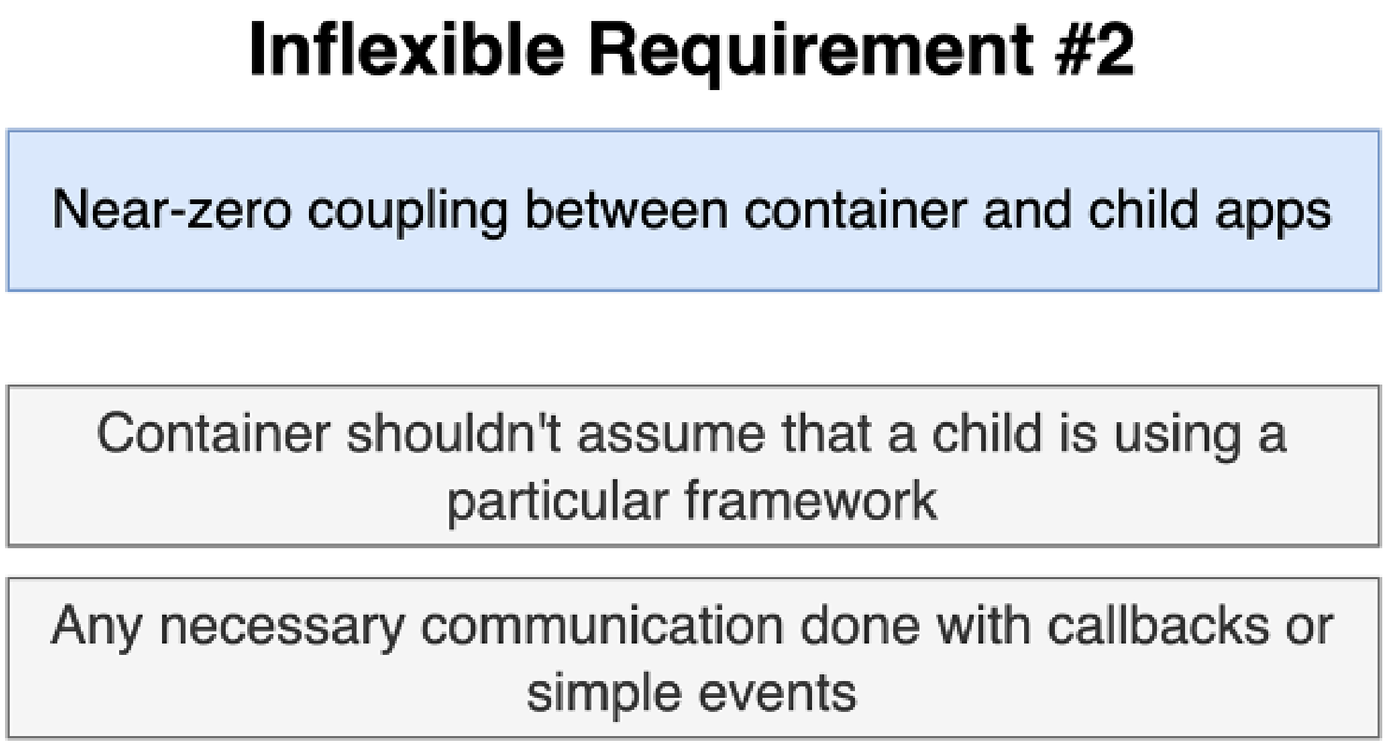
Không có sự liên kết giữa các dự án con





7 Yêu cầu dẫn đến lựa chọn kiến trúc

Gần như không có sự kết hợp giữa container và các ứng dụng con



7 Yêu cầu dẫn đến lựa chọn kiến trúc

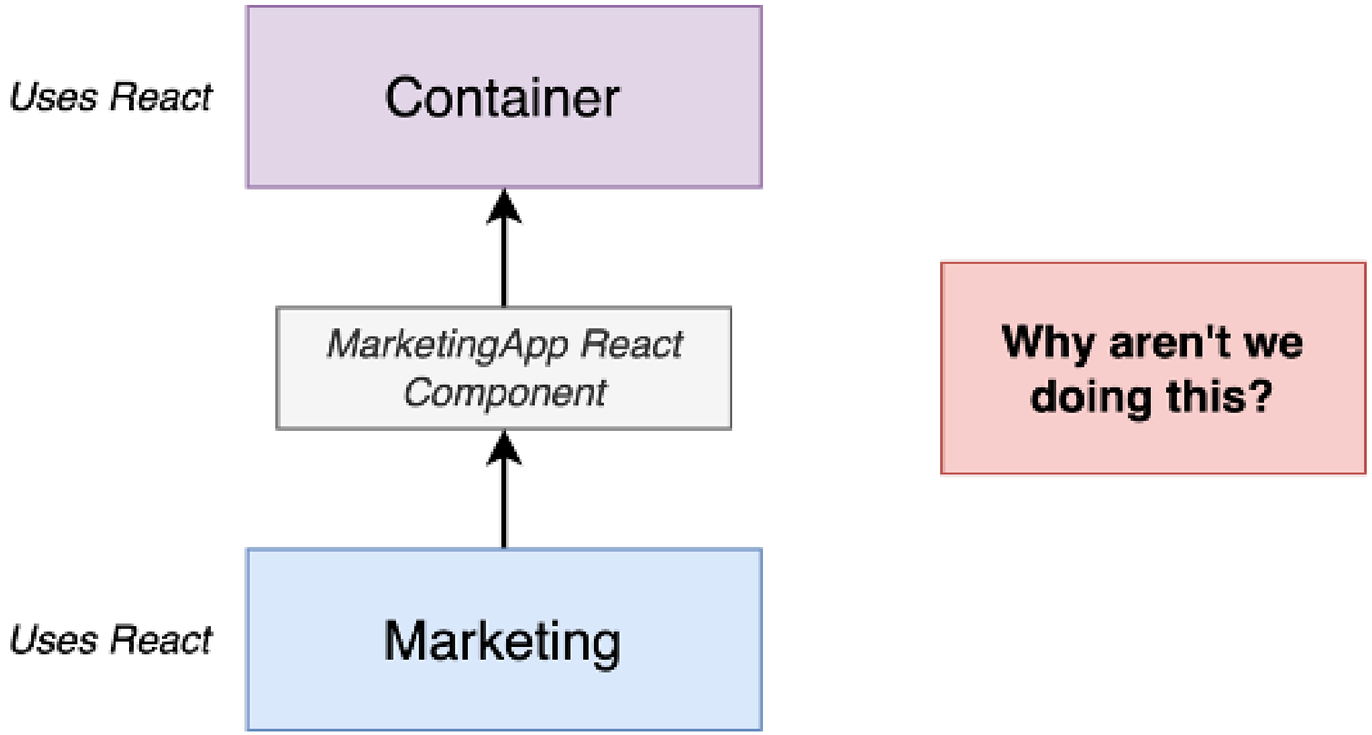
CSS từ một dự án không nên ảnh hưởng đến dự án khác

- Trong nhiều trường hợp, các dự án khi gộp lại với nhau sẽ xảy ra xung đột về các CSS Class, Selector,... Điều này sẽ gây ra khá nhiều vấn đề vì gần như việc đặt tên các Class, Selector là không thể đoán trước được.
- Chính vì vậy, có một số giải pháp và đặc biệt là giải pháp sử dụng CSS module để có thể đặt tên tuy giống nhau về mặt lý thuyết nhưng khi triển khai lên giao diện thì các Class, Selector đó sẽ không hề giống nhau, chúng sẽ được “mã hoá” một cách độc lập với các Class, Selector khác

7 Yêu cầu dẫn đến lựa chọn kiến trúc

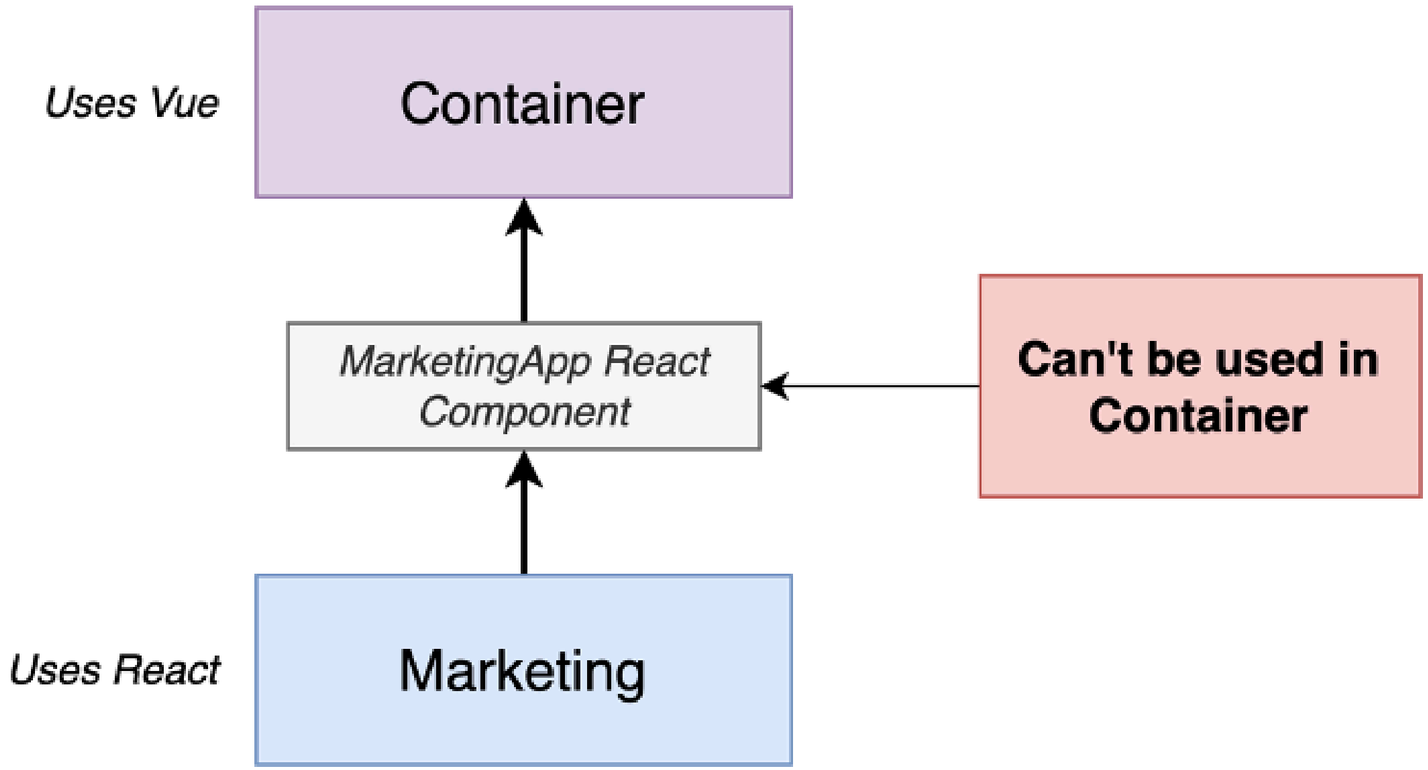
Tại sao phải sử dụng mount function để render UI?

- Giả sử ta Container sử dụng React và microfrontend Marketing cũng sử dụng React



7 Yêu cầu dẫn đến lựa chọn kiến trúc

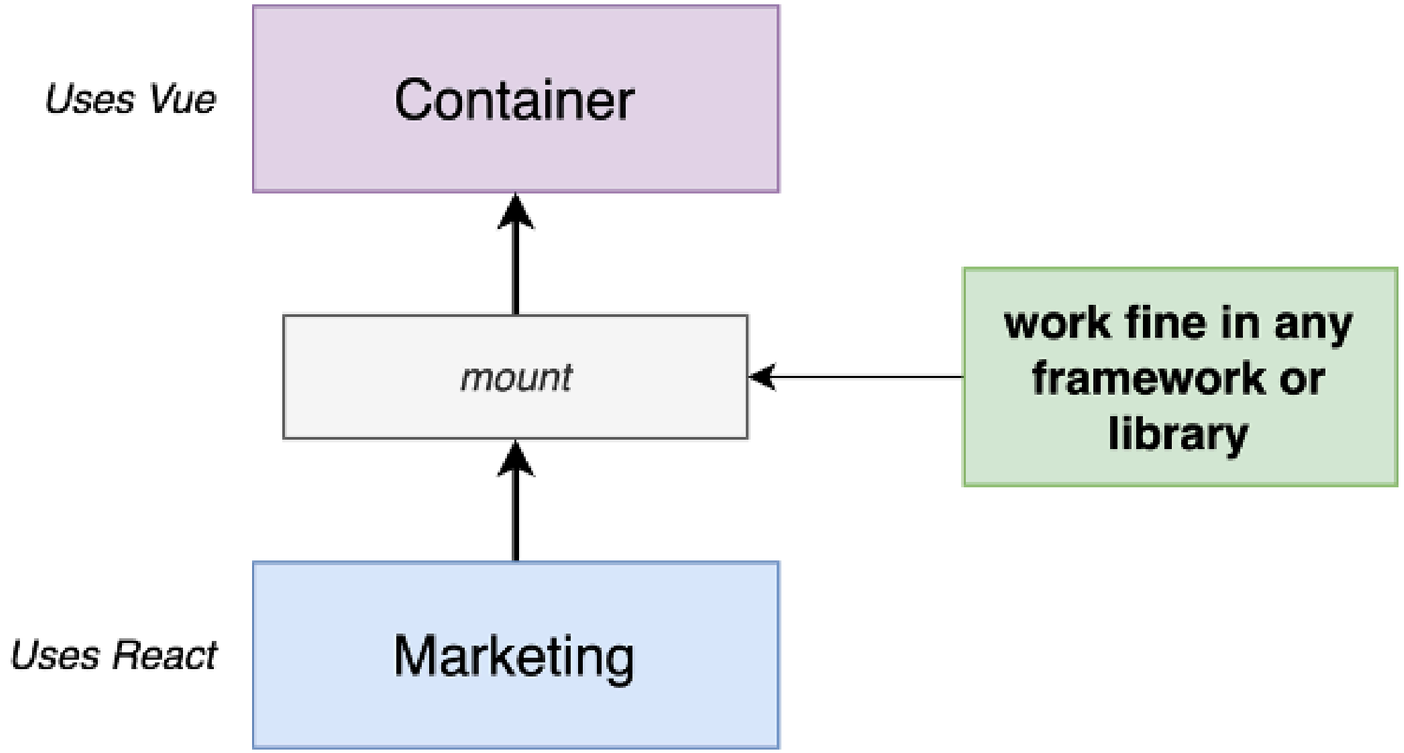
Tại sao phải sử dụng mount function để render UI?



7 Yêu cầu dẫn đến lựa chọn kiến trúc

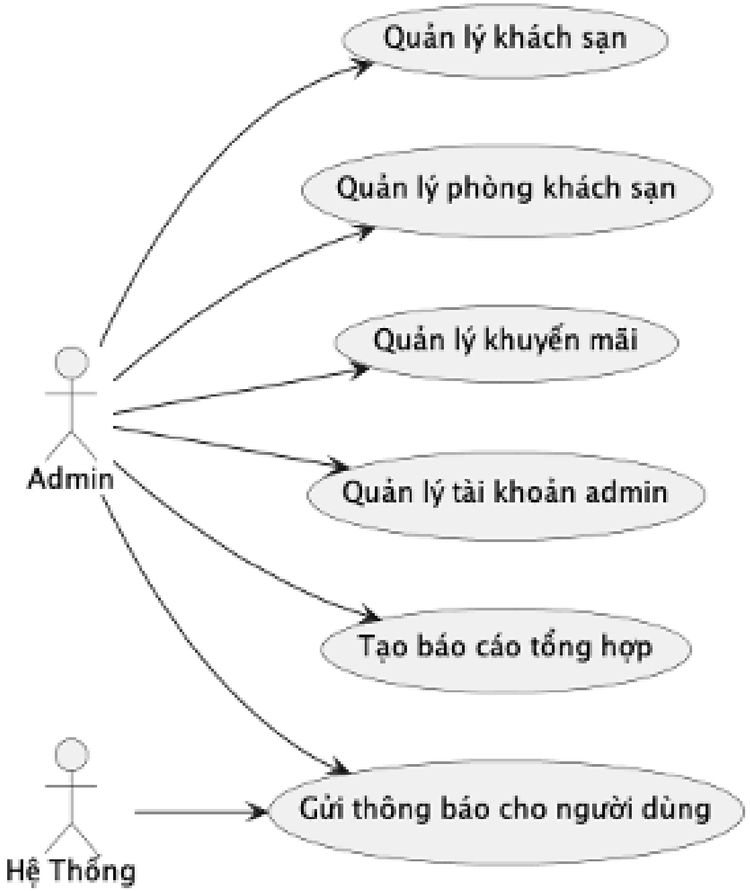
Tại sao phải sử dụng mount function để render UI?

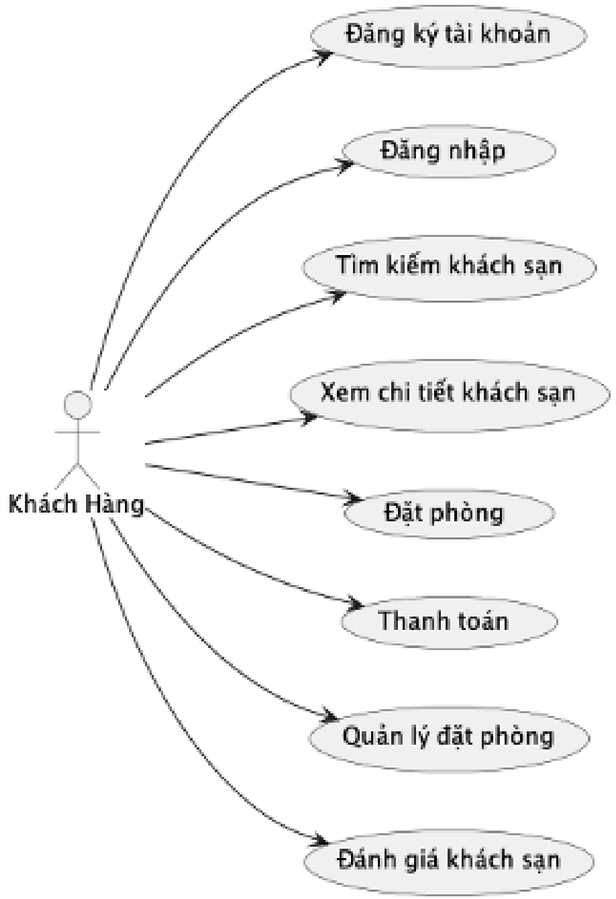
- Mount function sẽ chọn phần tử (HTML DOM) thông qua id/class ở trang web và render lên màn hình thông qua logic được định sẵn trong nó



8 Ứng dụng Microfrontend trong lĩnh vực đặt phòng khách sạn

Use case tổng quan





8 Ứng dụng Microfrontend trong lĩnh vực đặt phòng khách sạn

Các microfrontend chính

- Booking (ReactJS)
- User Profile (SolidJS)
- Marketing (VanillaJS)
- Payment (VanillaJS)
- Review (VanillaJS)
- Container (Host - ReactJS)

8 Ứng dụng Microfrontend trong lĩnh vực đặt phòng khách sạn

Demo

9 Kết luận

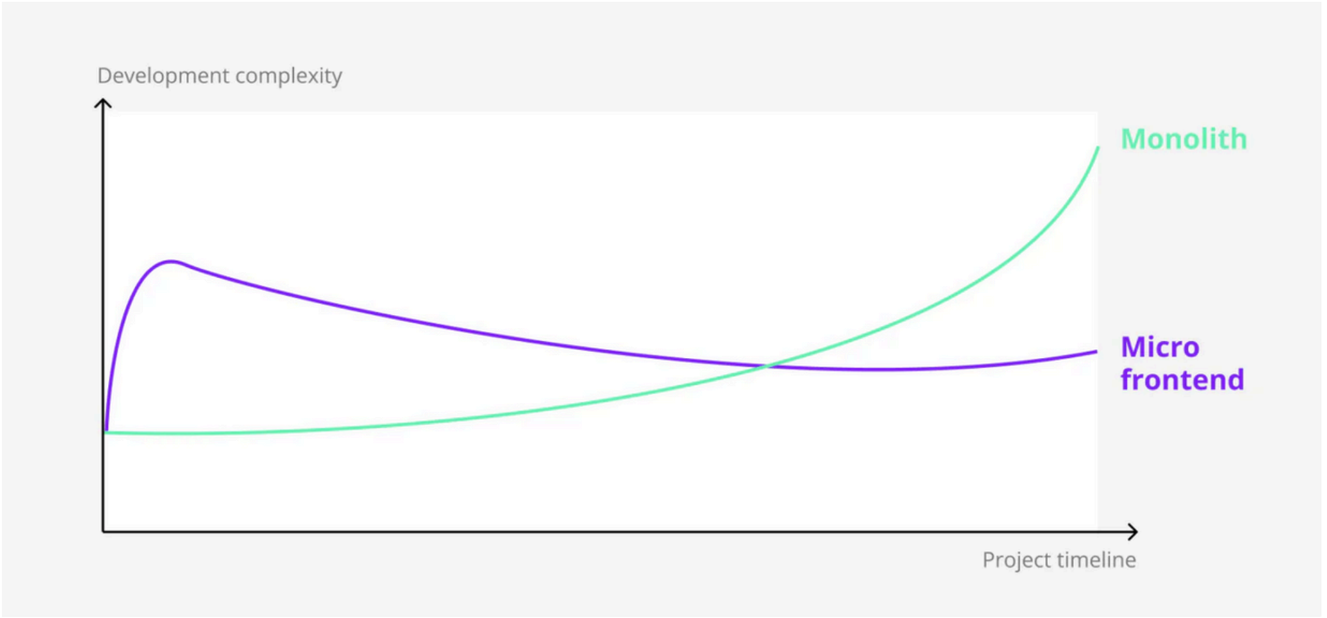
Lợi ích của Microfrontend trong phát triển ứng dụng

- Triển khai nhanh hơn và quản lý bản phát hành tốt hơn
- Nhiều nhóm có trách nhiệm khác nhau
- Tự do công nghệ
- Dễ dàng mở rộng quy mô
- Triển khai liên tục

9 Kết luận

Thách thức

- Kích thước tải trọng
- Giao tiếp giữa các trang
- Sự khác biệt về thiết kế
- Độ phức tạp về mặt vận hành



9 Kết luận

Phân công công việc

Giai đoạn	Tuần	Ngày	Công việc	Thành viên thực hiện
Lên kế hoạch	1, 2, 3, 4	30/9 - 20/10	- Tìm hiểu các nội dung liên quan đến đề tài - Lập bảng kế hoạch sắp tới - Chuẩn bị dàn ý nội dung	Kiệt, Mai
Đặc tả yêu cầu	5	21/10 - 26/10	- Tổng hợp chức năng và use case - Xây dựng các sơ đồ use case và đặc tả sơ đồ use case	Kiệt, Mai
Thiết kế	6	28/10 - 03/11	- Thiết kế giao diện	Mai
Xây dựng Backend	6, 7	28/10 - 10/11	- Xây dựng Backend cho toàn bộ ứng dụng	Kiệt
Xây dựng ứng dụng	8	11/11 - 17/11	- Xây dựng các Microfrontend: Booking	Mai
	9	18/11 - 25/11	- Xây dựng các Microfrontend: User Profile, Marketing	Kiệt
	10, 11	26/11 - 08/12	- Xây dựng các Microfrontend: Payment, Review	Kiệt
Triển khai ứng dụng	12	09/12 - 13/12	- Testing	Mai
	12	14/12 - 15/12	- Triển khai ứng dụng lên AWS S3 và AWS CloudFront	Kiệt
Báo cáo	13	16/12 - 23/12	- Viết báo cáo và hoàn thiện, chỉnh sửa (nếu có) - Chuẩn bị Slide thuyết trình	Kiệt, Mai

Thank you!

Q & A