**Motivation**

Nhiều bản build riêng biệt nên tạo thành một ứng dụng duy nhất. Các bản build riêng biệt này hoạt động như các container và có thể chia sẻ và sử dụng mã với nhau, tạo thành một ứng dụng thống nhất duy nhất.

Điều này thường được gọi là Micro-Frontends, nhưng không chỉ giới hạn ở đó.

**Một số concepts**

Phân biệt giữa các module cục bộ và từ xa. Các module cục bộ là các module thông thường là một phần của quá trình build hiện tại. Các module từ xa là các module không phải là một phần của quá trình build hiện tại nhưng được tải vào thời điểm chạy từ một container từ xa.

Việc tải các module từ xa được coi là một hoạt động bất đồng bộ. Khi sử dụng một module từ xa, các hoạt động bất đồng bộ này sẽ được đặt vào các hoạt động tải chunk tiếp theo nằm giữa module từ xa và điểm nhập chính. Không thể sử dụng một module từ xa mà không có hoạt động tải chunk.

Hoạt động tải chunk thường là gọi import(), nhưng các cấu trúc cũ như require.ensure hoặc require([...]) cũng được hỗ trợ.

Một container được tạo thông qua một entry container, nó cung cấp truy cập bất đồng bộ đến các module cụ thể. Truy cập được cung cấp được chia thành hai bước:

* tải module - asynchronous (bất đồng bộ)
* đánh giá module - synchronous (đồng bộ).

Bước 1 sẽ được thực hiện trong quá trình tải chunk. Bước 2 sẽ được thực hiện trong quá trình đánh giá module xen kẽ với các module khác (local và remote). Bằng cách này, thứ tự đánh giá không bị ảnh hưởng bởi việc chuyển đổi một module từ local sang remote hoặc ngược lại.

Có thể lồng các container. Các container có thể sử dụng các module từ các container khác. Tương tác tuần hoàn giữa các container cũng có thể xảy ra.

Mỗi bản build hoạt động như một container và cũng tiêu thụ các bản build khác dưới dạng container. Bằng cách này, mỗi bản build có thể truy cập bất kỳ module được expose nào khác bằng cách tải nó từ container của nó.

Các module được chia sẻ là các module có thể được ghi đè và được cung cấp dưới dạng ghi đè cho các container lồng nhau. Chúng thường trỏ đến cùng một module trong mỗi bản build, ví dụ: cùng một thư viện.

Tùy chọn packageName cho phép đặt tên gói để tìm kiếm một requiredVersion. Nó được suy luận tự động cho các yêu cầu module theo mặc định, đặt requiredVersion thành false khi cần tắt suy luận tự động.

**Use cases**

**Separate builds per page**

Mỗi trang của một Ứng dụng Trang Đơn (Single Page Application - SPA) được expose từ một bản build container trong một bản build riêng biệt. Shell ứng dụng cũng là một bản build riêng biệt tham chiếu đến tất cả các trang dưới dạng các module từ xa. Bằng cách này, mỗi trang có thể được triển khai riêng biệt. Shell ứng dụng được triển khai khi các route được cập nhật hoặc các route mới được thêm vào. Shell ứng dụng định nghĩa các thư viện được sử dụng chung dưới dạng các module được chia sẻ để tránh sự trùng lặp của chúng trong các bản build trang.

**Components library as container**

Nhiều ứng dụng chia sẻ một thư viện thành phần chung có thể được xây dựng dưới dạng một container với mỗi thành phần được expose. Mỗi ứng dụng tiêu thụ các thành phần từ container thư viện thành phần. Các thay đổi đối với thư viện thành phần có thể được triển khai riêng biệt mà không cần phải triển khai lại tất cả các ứng dụng. Ứng dụng tự động sử dụng phiên bản mới nhất của thư viện thành phần.

**Dynamic Remote Containers**

Interface container hỗ trợ các phương thức get và init. init là một phương thức tương thích với async được gọi với một đối số: đối tượng phạm vi chia sẻ. Đối tượng này được sử dụng làm phạm vi chia sẻ trong container từ xa và được điền bằng các module được cung cấp từ một host. Nó có thể được tận dụng để kết nối các container từ xa với một container host một cách động tại thời điểm chạy.

A screen shot of a computer code

Description automatically generated

Container cố gắng cung cấp các module được chia sẻ, nhưng nếu module được chia sẻ đã được sử dụng, một cảnh báo sẽ được hiển thị và module được chia sẻ được cung cấp sẽ bị bỏ qua. Container vẫn có thể sử dụng nó làm fallback.

Bằng cách này, bạn có thể tải động một bài kiểm tra A/B cung cấp một phiên bản khác của một module được chia sẻ.

Ví dụ:

A computer screen shot of a program

Description automatically generated

**Promise Based Dynamic Remotes**

Thông thường, các remote được cấu hình bằng URL như trong ví dụ này:

A computer screen shot of a program code

Description automatically generated

Nhưng bạn cũng có thể truyền vào một promise cho remote này, promise này sẽ được giải quyết tại thời điểm chạy. Bạn nên giải quyết promise này với bất kỳ module nào phù hợp với interface get/init được mô tả ở trên. Ví dụ: nếu bạn muốn truyền vào phiên bản của một module liên bang mà bạn nên sử dụng, thông qua một tham số truy vấn.

**Dynamic Public Path**

**Offer a host API to set the publicPath**

Người ta có thể cho phép host thiết lập publicPath của một module từ xa tại thời điểm chạy bằng cách expose một phương thức từ module từ xa đó.

Cách tiếp cận này đặc biệt hữu ích khi bạn gắn các ứng dụng con được triển khai độc lập vào sub path của domain host.

Scenario:

Giả sử bạn có một ứng dụng host được lưu trữ trên https://my-host.com/app/\* và một ứng dụng con được lưu trữ trên https://foo-app.com. Ứng dụng con cũng được gắn trên domain host, do đó, https://foo-app.com được mong đợi có thể truy cập được thông qua https://my-host.com/app/foo-app và các yêu cầu https://my-host.com/app/foo-app/\* được chuyển hướng đến https://foo-app.com/\* thông qua một proxy.