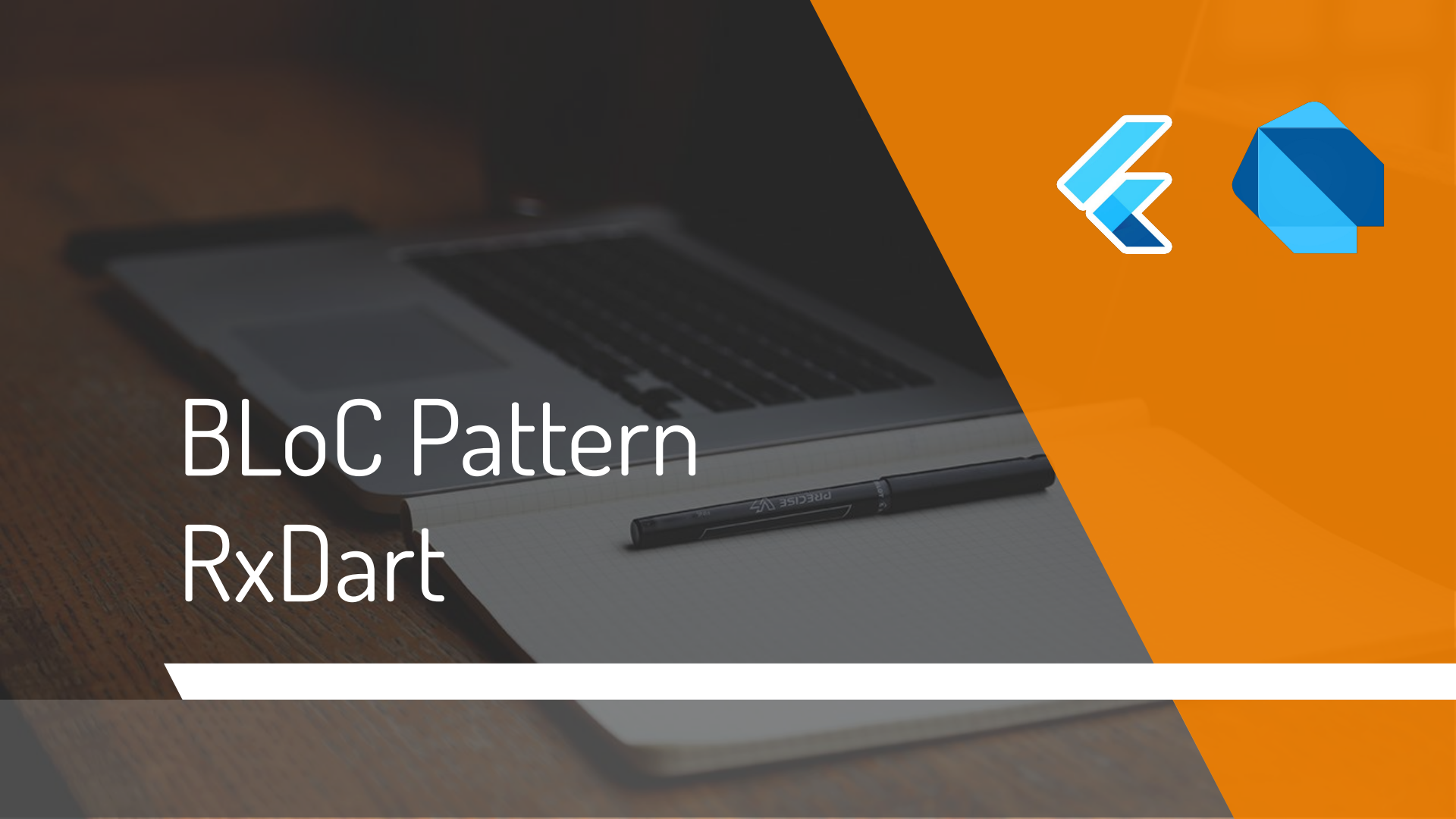


BLoC Pattern
RxDart

HELLO!

My name's Huy.

I’m a Software Engineer, I love special things and constantly learn.

Mục Lục

1.
State

Quản lý trạng thái của Widget

Giới Thiệu

State là gì ?

State là những thông tin có thể được đọc một cách đồng bộ khi Widget được xây dựng. Có thể thay đổi trong suốt vòng đời của Widget.

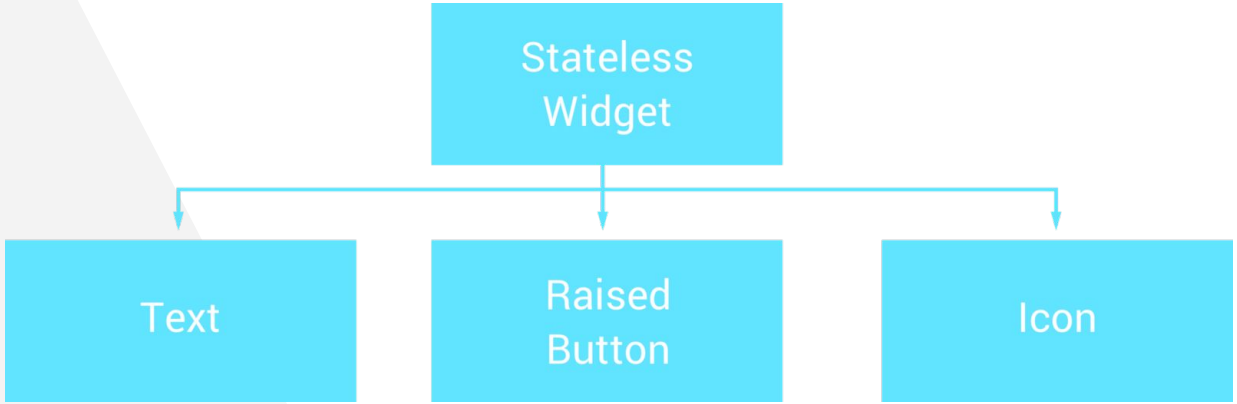
- **Stateless Widget** : không mang lại trạng thái, một khi đã được khởi tạo, sẽ không bị thay đổi cho đến khi được khởi tạo lại.
- **Stateful Widget** : mang trạng thái, có thể thay đổi trong suốt vòng đời mà không cần phải khởi tạo lại.

https://blog.geekyants.com/state-management-in-flutter-7df833e6f3bd

https://medium.com/@agungsurya/basic-state-management-in-google-flutter-6ee73608f96d

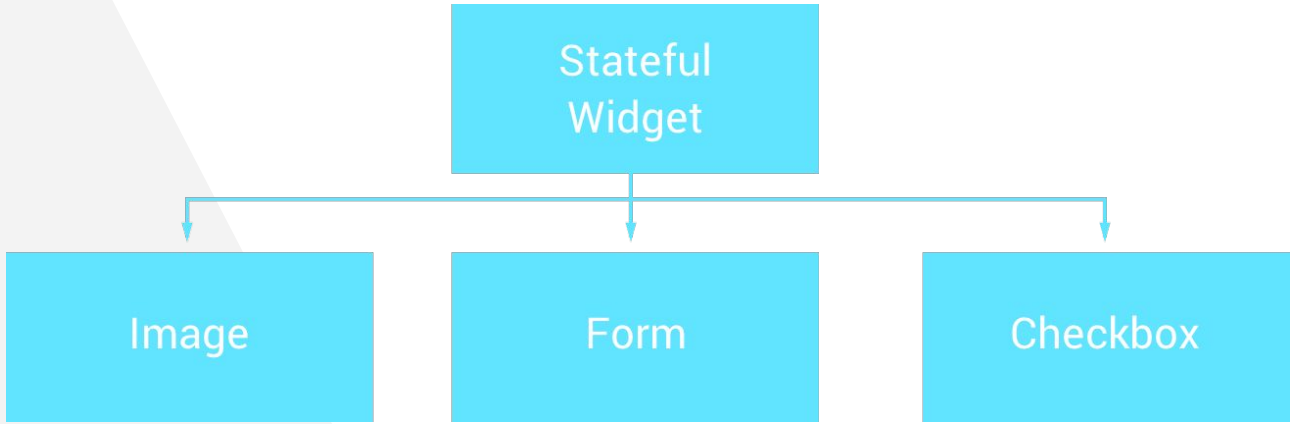
Stateless Widget

- Không có state.
- Không nhận sự thay đổi bên trong nó.
- Nhận dữ liệu và hiển thị một cách thụ động.
- Không sinh ra event khi tương tác để có thể tự render lại.

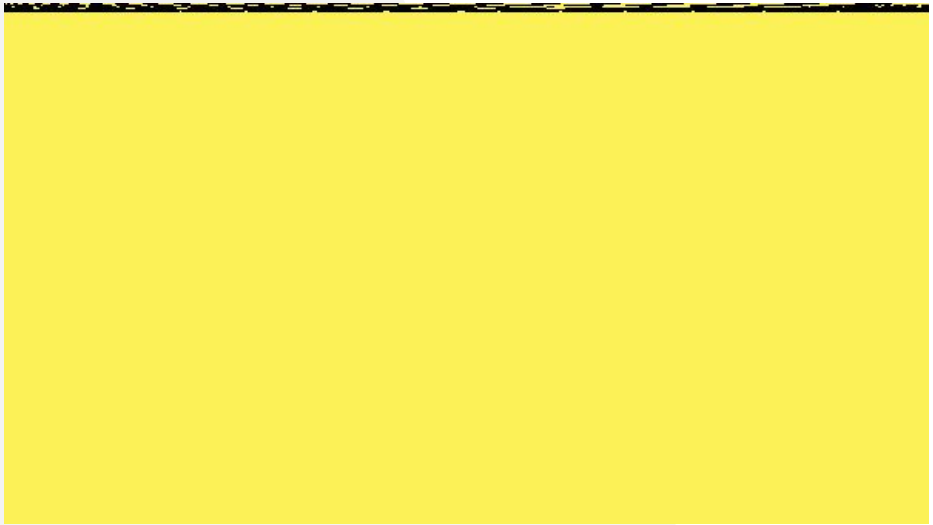


Stateful Widget

- Có chứa state.
- Cho phép sự thay đổi bên trong nó.
- Bất cứ khi nào một event được sinh ra nó sẽ lấy giá trị từ event đó để render lại **Widget.**



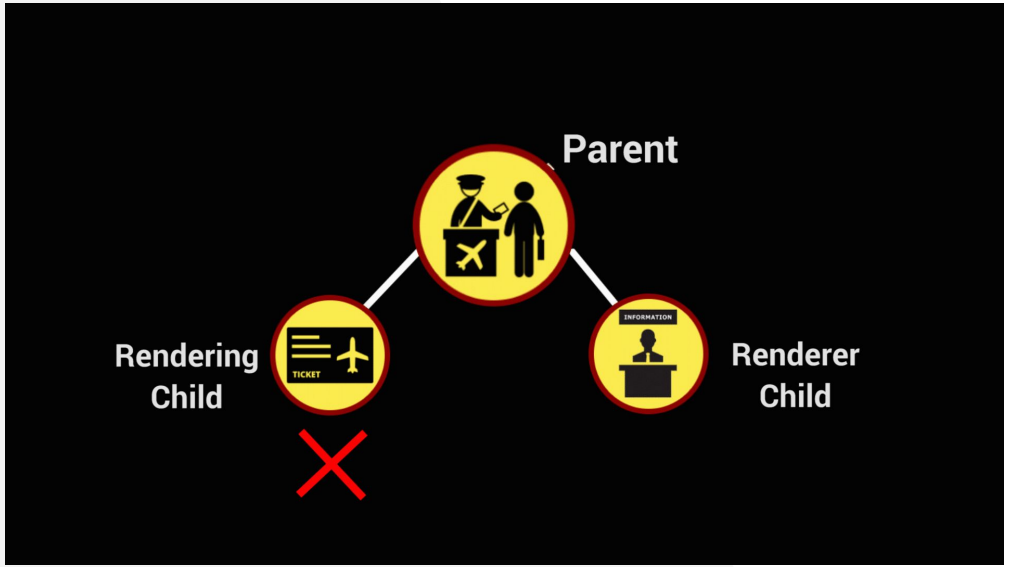
Render từ Parent đến Child



Trên một cây thành phần Widget, khi setState() được gọi từ các thành phần cha, nó sẽ vẽ lại tất cả thành phần con cháu.

Điều này gây ra sự cập nhật không cần thiết cho những thành phần con không thay đổi trạng thái.

Render từ Child đến Child



Trên một cây thành phần Widget, nếu hai thành phần con của một thành phần cha không cùng chung một lớp với thành phần cha, khi cập nhật một thành phần con thì không thể tác động được đến thành phần con khác.Trừ khi phải cập nhật thành phần cha của chúng.

DEMO

Nhưng kĩ thuật này có nhược điểm như vậy, tại sao lại cứ phụ thuộc hoàn toàn vào hàm setState () ?

2.
BLoC

Business Logic Components

Giới Thiệu

BloC (Business Logic Components) :

Là một thành phần tách logic nghiệp vụ của ứng dụng ra khỏi giao diện người dùng thông qua việc sử dụng **Streams.**

Được xây dựng dựa trên **Streams** và vận hành bởi **Inherited Widgets.**

http://flutterdevs.com/blog/bloc-pattern-in-flutter-part-1/

https://www.didierboelens.com/2018/08/reactive-programming---streams---bloc/

Vì sao lại là BLoC ?

Là một nhà phát triển, cần phải quan tâm :

- Trạng thái ứng dụng tại bất kì thời điểm nào.
- Dễ dàng kiểm tra mọi trường hợp để đảm bảo ứng dụng phản hồi phù hợp.
- Ghi lại mọi tương tác của người dùng trong ứng dụng để có thể đưa ra quyết định dựa trên dữ liệu.
- Hoạt động hiệu quả nhất có thể, tái sử dụng các thành phần trong ứng dụng và trên các ứng dụng khác.
- ...

Streams là gì ?

Là một chuỗi các sự kiện không đồng bộ.

Là một dòng chảy liên tục hoặc kế tiếp của bất cứ điều gì, có thể là: value, event, object, collection, map, error, ...

Type Streams

Single-subscription Streams

Là loại **Stream** chỉ cho phép tạo duy nhất 1 lần lắng nghe.

Lưu ý: Đừng cố gắng tạo thêm lần lắng nghe nào khác trỏ vào **Stream** này, vì nó sẽ huỷ đi lần lắng nghe đầu tiên.

Broadcast Streams

Là loại **Stream** cho phép tạo nhiều lần lắng nghe.

Lưu ý: Có thể tạo rất nhiều lần lắng nghe vào loại **Stream** này và nó chỉ lắng nghe các sự kiện từ lúc lắng nghe đó được tạo ra.

BLoC Rules:

- **Stream** là nơi truyền tải dữ liệu. (Ống nước)
- Để điều khiển được stream, ta thường dùng tới **StreamController**.
- **StreamTransformer** là nơi dữ liệu đầu vào sẽ được biến đổi.
- **StreamBuilder** là một phương thức lấy stream làm đầu vào và cung cấp một nơi để xây dựng lại mỗi khi có một giá trị mới của stream được đẩy ra.
- **sink**, thuộc tính để đẩy dữ liệu đầu vào. (**StreamSink**)
- **stream**, thuộc tính cung cấp lối ra của **Stream**.
- **StreamSubscription** cung cấp các event cho lần lắng nghe và giữ các lần lắng nghe được sử dụng để xử lý các sự kiện.

RxDart

RxDart là một thư viện **Reactive Programming** cho Dart, dựa trên ReactiveX.

Dart đi kèm với một API Streams, thay vì cố gắng cung cấp một giải pháp thay thế cho API này, RxDart thêm chức năng lên trên nó. Do đó 1 số ngữ cảnh sẽ thay đổi :

Dart	RxDart
Stream	Observable
StreamController	Subject

https://pub.dev/packages/rxdart

https://medium.com/flutter-community/why-use-rxdart-and-how-we-can-use-with-bloc-pattern-in-flutter-a64ca2c7c52d

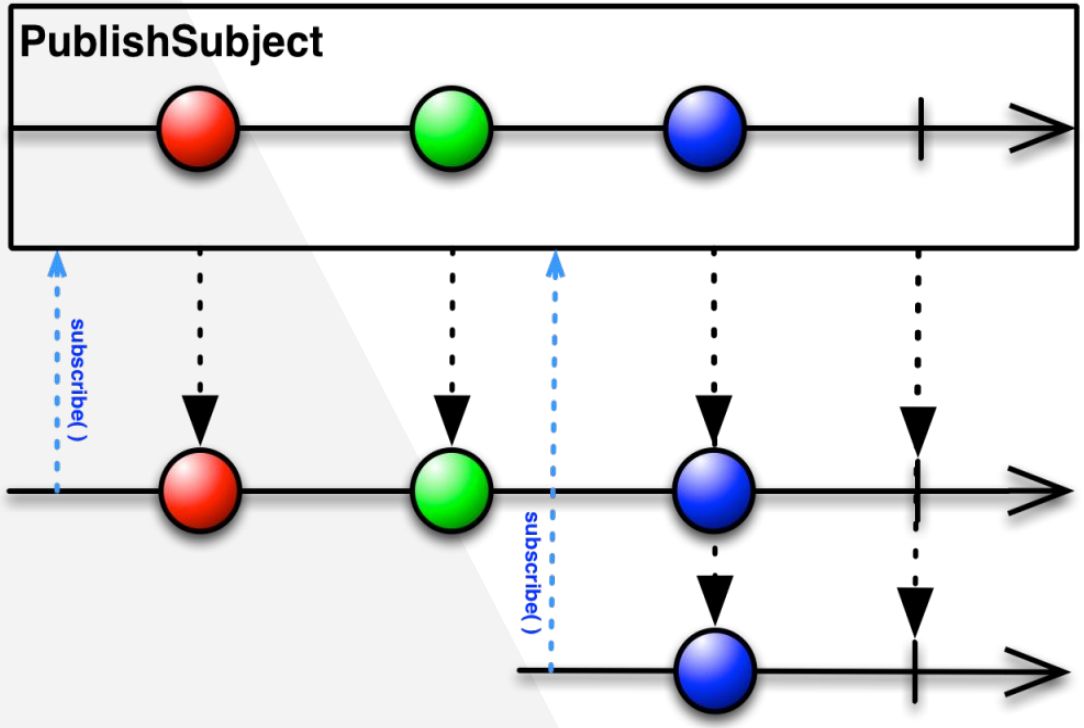
Observable

Là một đối tượng mở rộng của lớp **Stream**, có 2 điều ta cần lưu ý:

Tất cả các phương thức được định nghĩa trên lớp Stream cũng tồn tại trên Observable của RxDart.

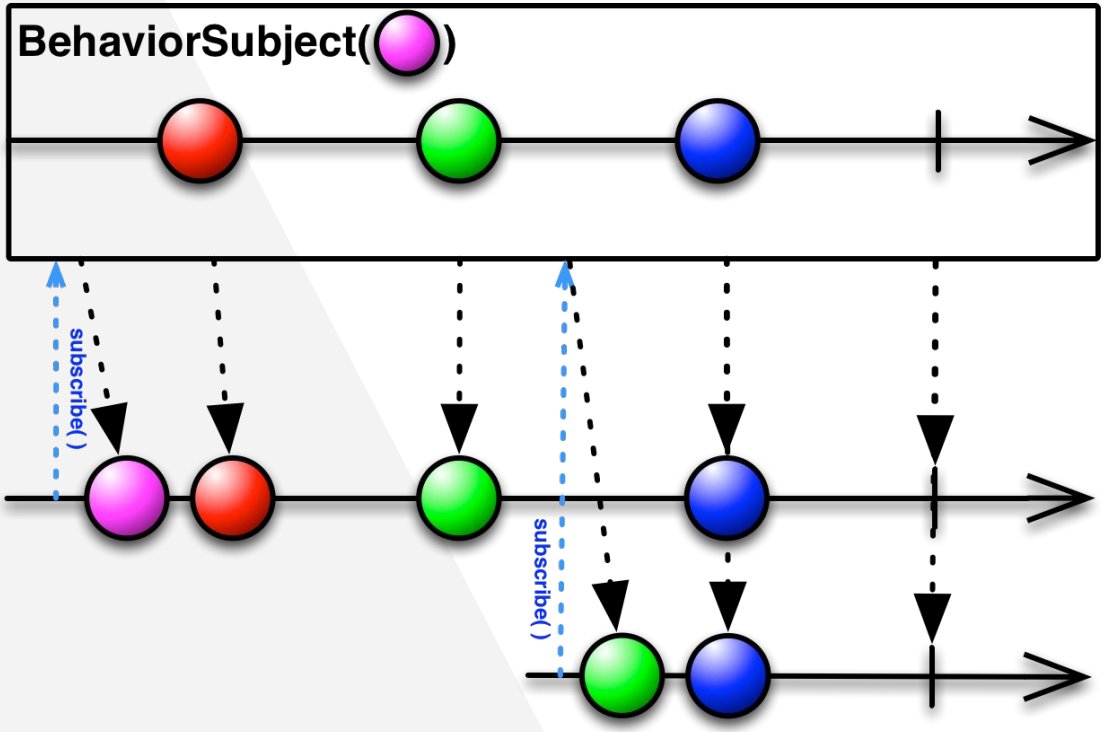
Tất cả các Observable có thể được chuyển đến bất kỳ API nào với **Stream** làm đầu vào.

PublishSubject



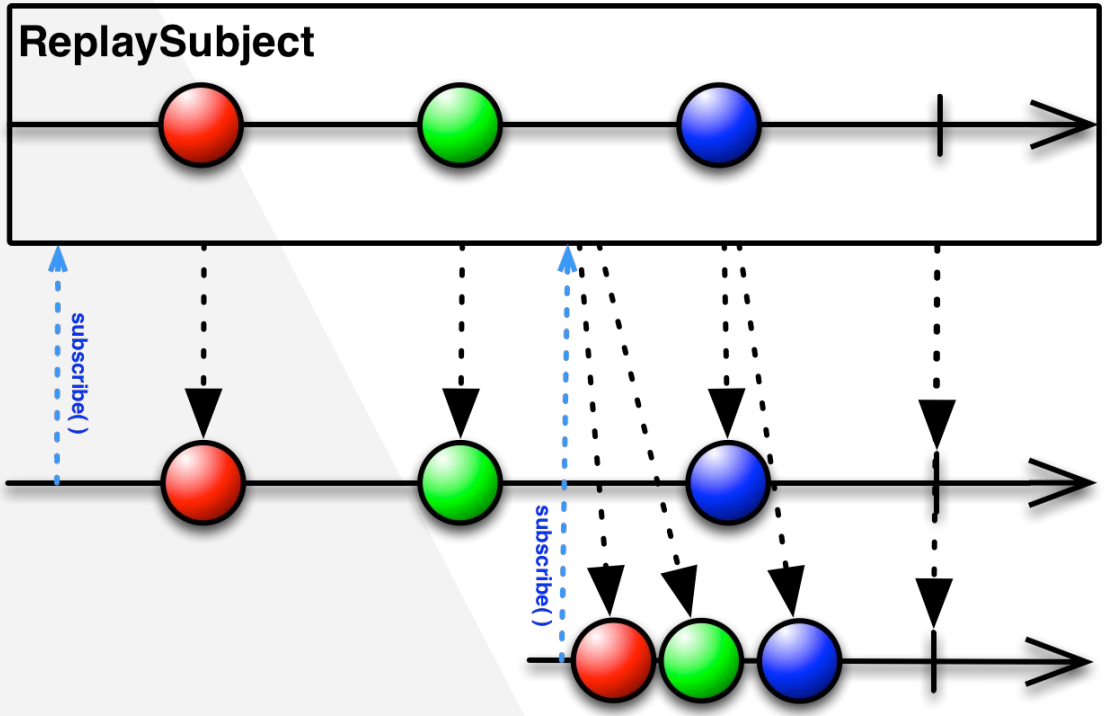
Là một Broadcast StreamController, với 1 sự khác biệt là nó sẽ trả về 1 Observable thay vì Stream.

BehaviorSubject



Là một Broadcast StreamController, với 1 sự khác biệt là nó sẽ trả về 1 Observable thay vì Stream.

ReplaySubject



Là một Broadcast StreamController, với 1 sự khác biệt là nó sẽ trả về 1 Observable thay vì Stream.

LƯU Ý: Hãy luôn giải phóng tất cả các Resources khi chúng không còn cần thiết !!

DEMO

THANKS!

Any questions?

You can find me at **thaihuycr@gmail.com**