Reactive Programming RxJava

What is Reactive Programming?

Reactive programming is programming with asynchronous data streams. Event buses or your typical click events are really an asynchronous event stream, on which you can observe and do some side effects. Reactive is that idea on steroids. You are able to create data streams of anything, not just from click and hover events. Streams are cheap and ubiquitous, anything can be a stream: variables, user inputs, properties, caches, data structures, etc. For example, imagine your Twitter feed would be a data stream in the same fashion that click events are. You can listen to that stream and react accordingly.

What is Stream?

A stream is a sequence of ongoing events ordered in time. It can emit three different things: a value (of some type), an error, or a "completed" signal.

Is RxJava Following The “Push” Or “Pull” Pattern?

In RxJava new data is being “pushed” to observers.

What’s The Difference Between OnNext(), OnComplete() And OnError()?

These are the callbacks an Observable / Flowable will receive. The first one is called for each emission of the Observable / Flowable (e.g. zero to infinity times). onComplete() and onError() are mutually exclusive – only ONE of them will be called at most once. In other words a stream cannot complete and error out at the same time.

How Many Times Can Each Of The OnNext(), OnComplete() And OnError() Called?

onNext() – from zero between infinite number of times
onComplete() – maximum once per stream
onError() – maximum once per stream

When Does An Observable Start Emitting Items?

There’s two types of Observables – Cold and Hot. Cold ones perform work (and subsequently emit items) only once someone is listening for it (e.g. someone has subscribed to them). Hot observables perform work and emit items regardless if there are any observers or not.

What’s The Difference Between A COLD And HOT Observables?

They start emitting items differently.

Cold observables are created multiple times and each instance can be triggered on it’s own. Hot observables are like a “stream” of ongoing events – observers can come and

go, but the stream is created ones and just goes on.

Can You Transform A COLD Observable To A HOT One And Vice-Versa?

One way to make a Cold observable Hot is by using publish().connect(). publish() converts the Cold observable to a ConnectableObservable, which pretty much behaves like a Hot one. Once triggered with the .connect() operator, it'll publish events regardless if there are any subscribers.

Another way to transform a Cold observable to a Hot one is by wrapping it with a Subject. The Subject subscribes to the Cold observable immediately and exposes itself as an Observable to future subscribers. Again, the work is performed regardless whether there are any subscribers … and on the other hand multiple subscribers to the Subject won't trigger the initial work multiple times.

What's A Scheduler? Why Does RxJava Use Schedulers?

By default RxJava is single-threaded – all operations are executed on a single thread. Schedulers are the means to switch the execution to a different thread. They're also an abstraction over the concept of "time", which is needed for time-sensitive operations (delay(), timeout(), buffer(), window(), etc).

https://veskoiliev.com/40-rxjava-interview-questions-and-answers/

Dependency Injection

What is Dependency Injection?

Dependency injection makes it easy to create loosely coupled components, which typically means that components consume functionality defined by interfaces without having any first-hand knowledge of which implementation classes are being used.

Dependency injection makes it easier to change the behaviour of an application by changing the components that implement the interfaces that define application features. It also results in components that are easier to isolate for unit testing.

What is Dagger?

Dagger is a dependency injection library for Android. It is used for injecting objects into other objects, such as activities and fragments. This allows for a more modular codebase, and can help to make code more testable and easier to maintain.