

RxJava

Reactive Extensions for the JVM

Part 1: Introduction to Rx


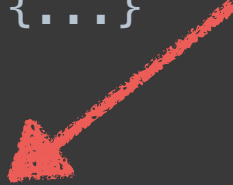
Part 2: Workshop

Java is **functional**!



```
Optional<Asset> findAsset(AssetId assetId) {...}
```

```
Optional<URI> getShareURI(AssetId assetId,  
                          Function<Adaptor, Boolean> supportSharing,  
                          Function<Adaptor, URI> generateURI) {  
  
    return findAsset(assetId)  
        .flatMap(asset ->  
            Stream.concat(Stream.of(primaryAdaptor), secondaryAdaptors.stream())  
                .filter(withExternalReference(asset))  
                .filter(supportSharing::apply)  
                .map(generateURI::apply)  
                .findFirst());  
        }  
}
```



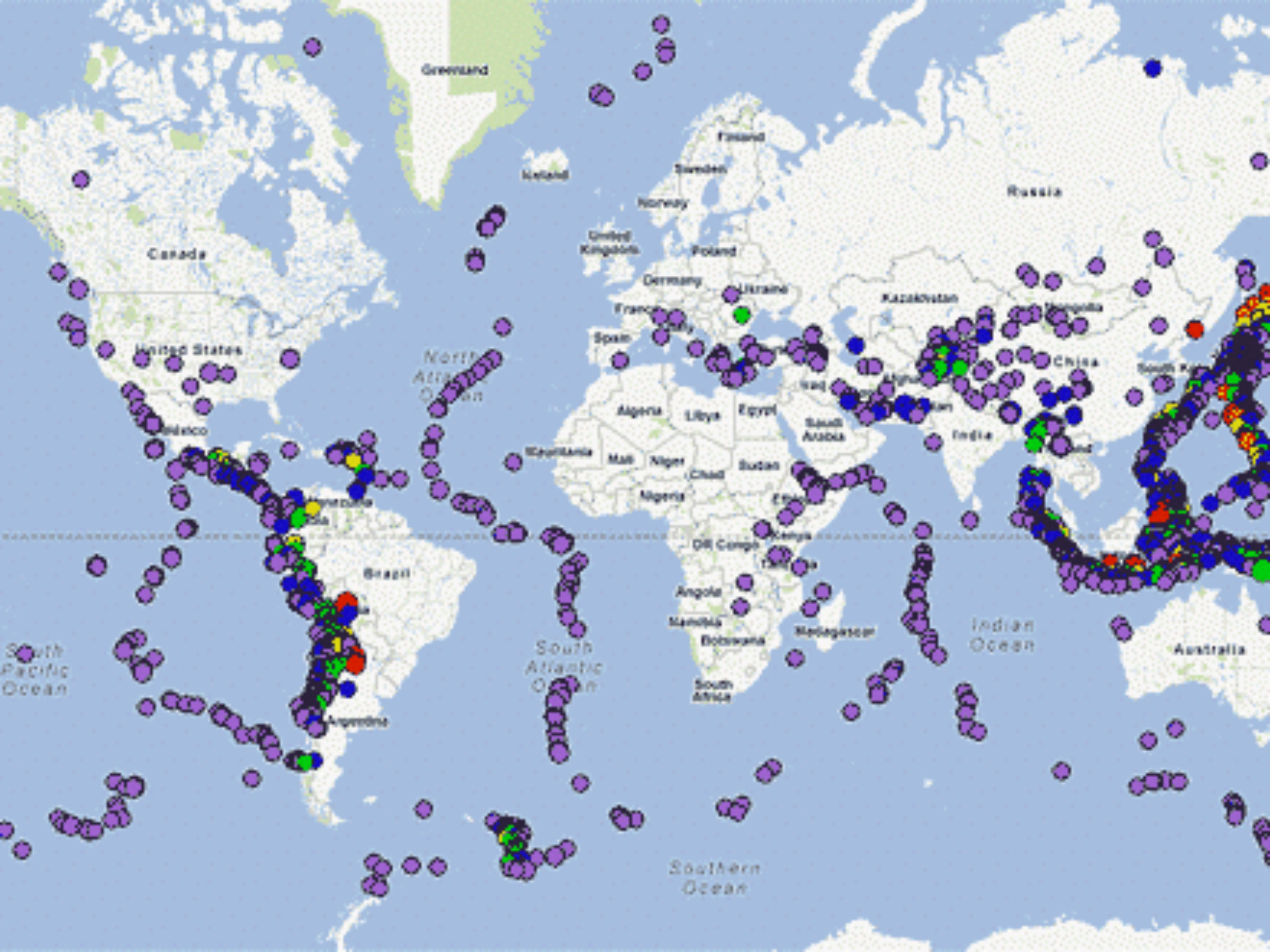
A man with a beard and short hair, wearing a green t-shirt with a large sunflower graphic, stands in front of a brick wall. He has his arms outstretched and is speaking. In front of him is a silver laptop with the Apple logo on the back, and a red folder or book is open on a stand. A large black screen is visible behind him to the right.

**«We have to start to think
in terms of streams»**

Eric Meijer



See streams
everywhere



**These streams do
not wait for you!**



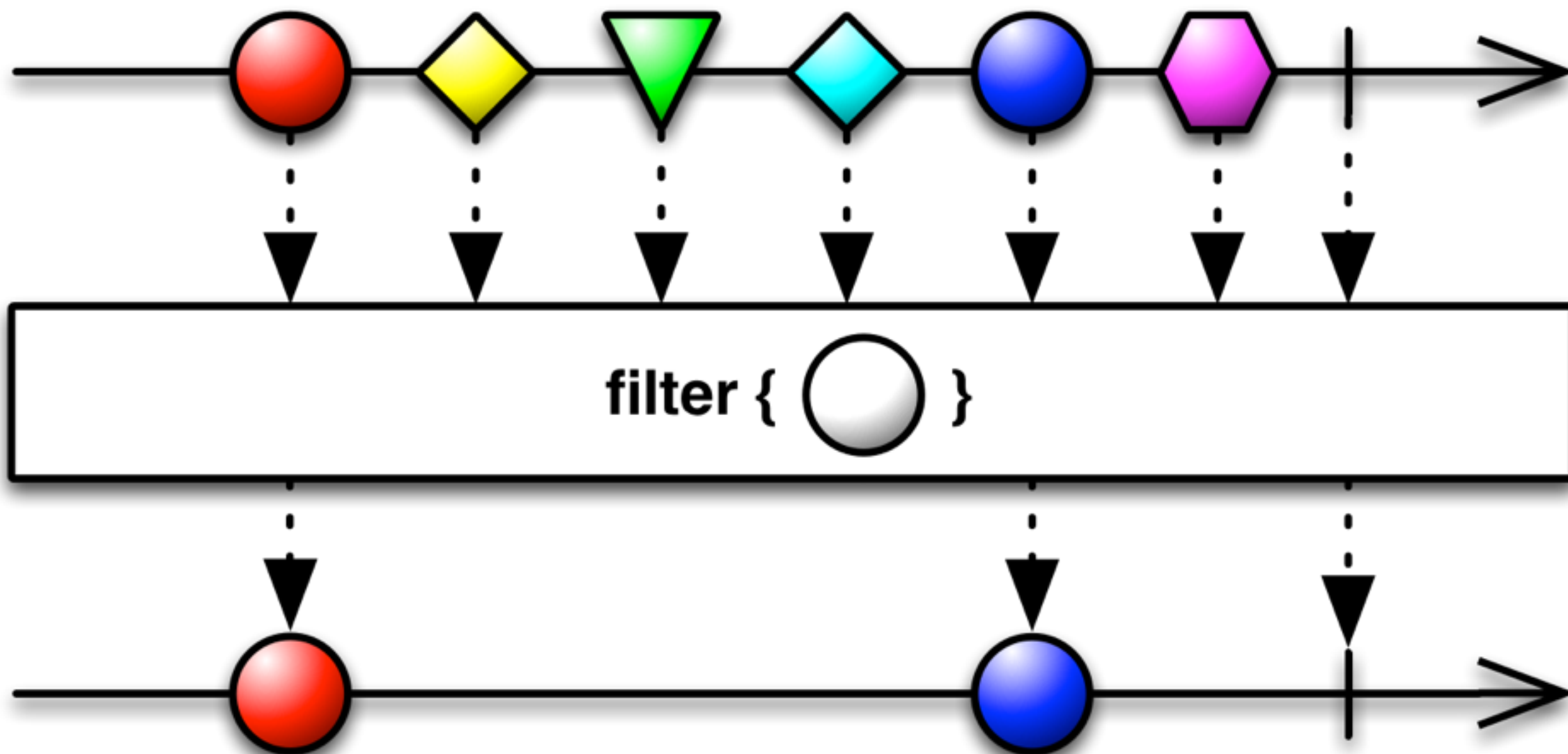
Observable streams!

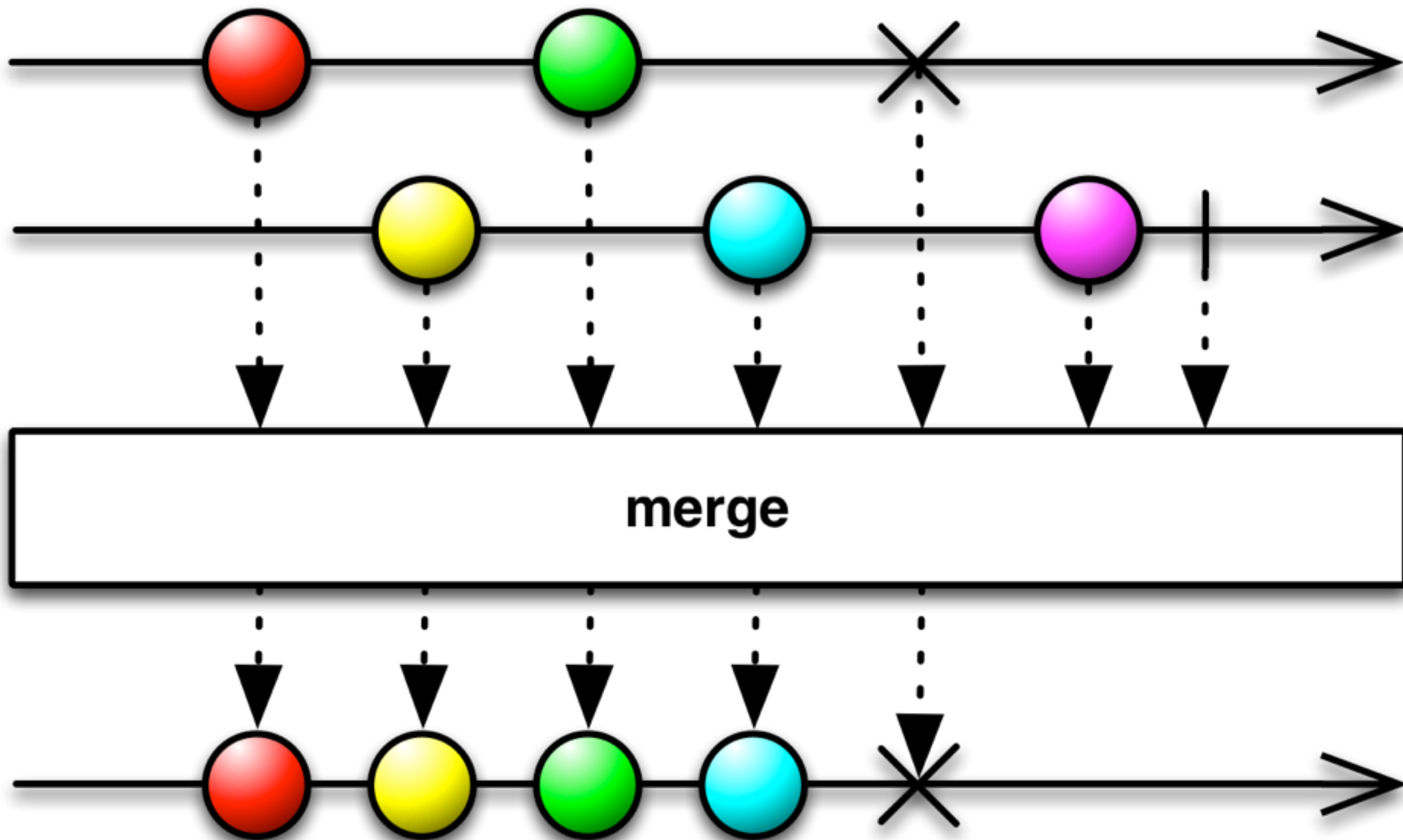
Iterable «Pull»

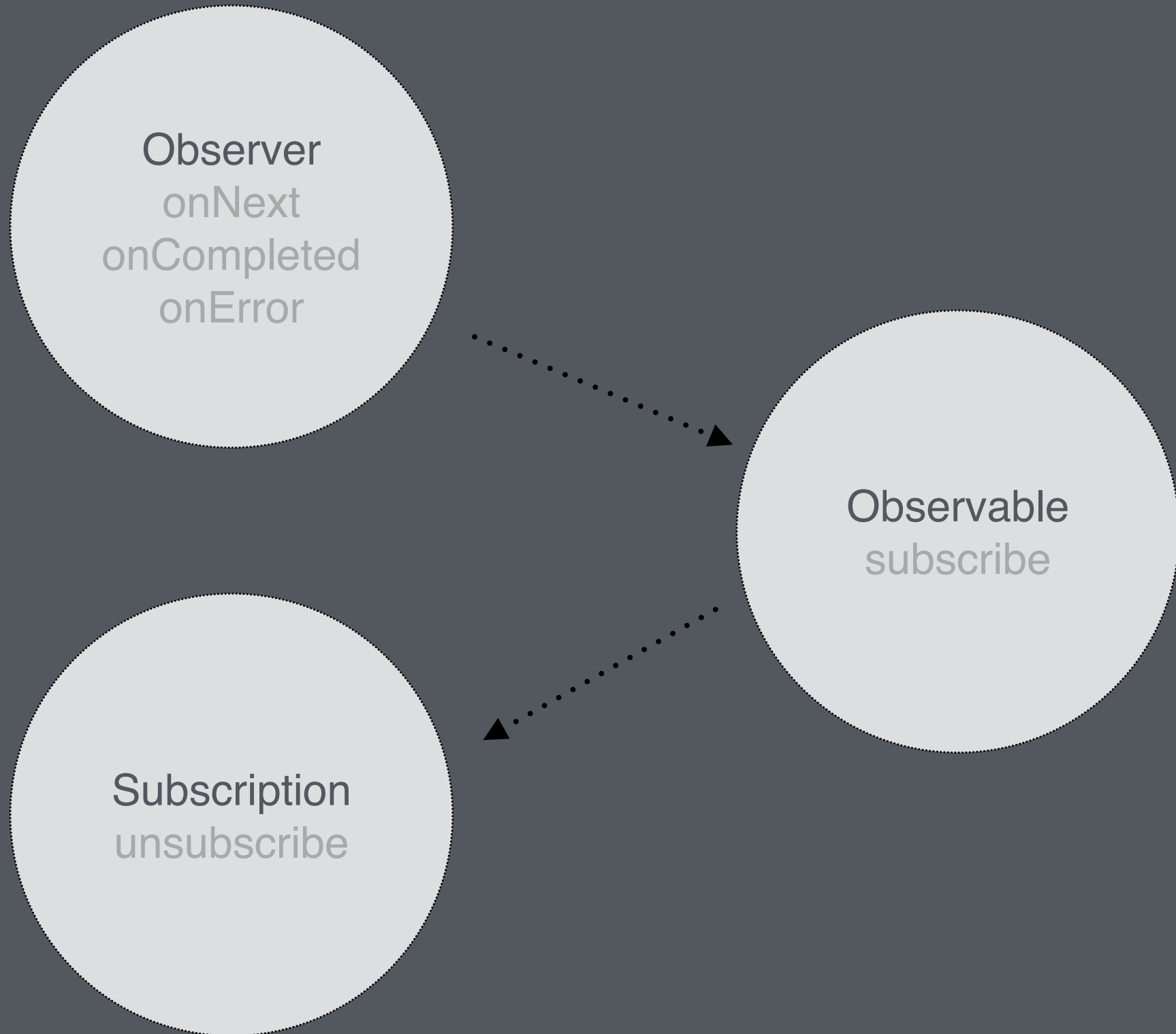
```
getJava8Stream()  
  .skip(10)  
  .take(5)  
  .map(s -> s + " transformed")  
  .forEach(System.out::println);
```

Observable «Push»

```
getObservableStream()  
  .skip(10)  
  .take(5)  
  .map(s -> s + " transformed")  
  .forEach(System.out::println);
```





Event-based
UI

Rx =
abstractions
+
«glue»



Client



Micro

Async
computation
Error handling
Timeouts



**Show me some
code!**

Part 2: RxSnake

`git@github.com:mlundela/rx-demo.git`