Reactive Android Intro: issues, examples and how to succeed with RxJava and Cascade

4/28/2015

Lucia Payo and Paul Houghton
Futurice GmbH

futurice

Tampere
Helsinki
Berlin
London
Zug

- What is reactive programming?
- Concepts
- Let’s Code!

- What is reactive programming?
- Concepts
- Let’s Code!

What is reactive?

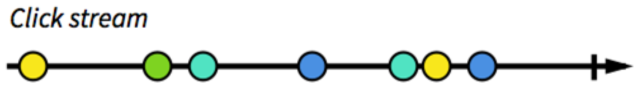
- Show a dialog after a double click

What is reactive?

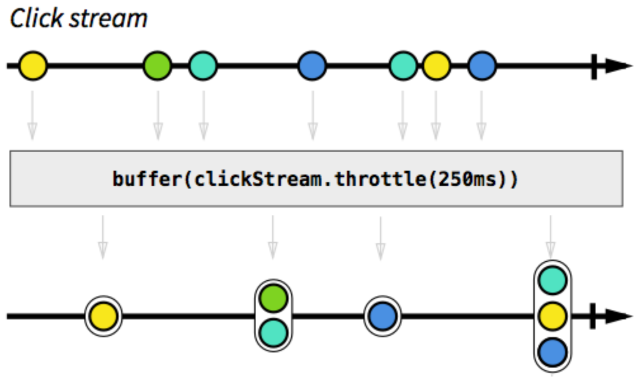
- Show a dialog after a double click

- Multiple clicks == double clicks

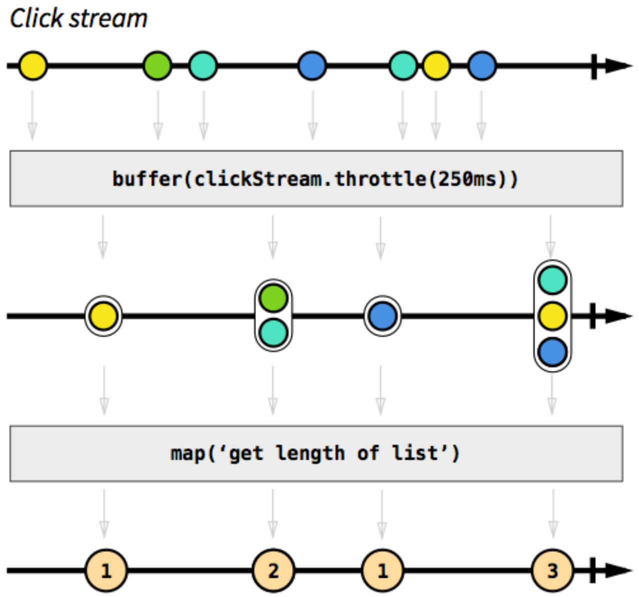
What is reactive?



What is reactive?

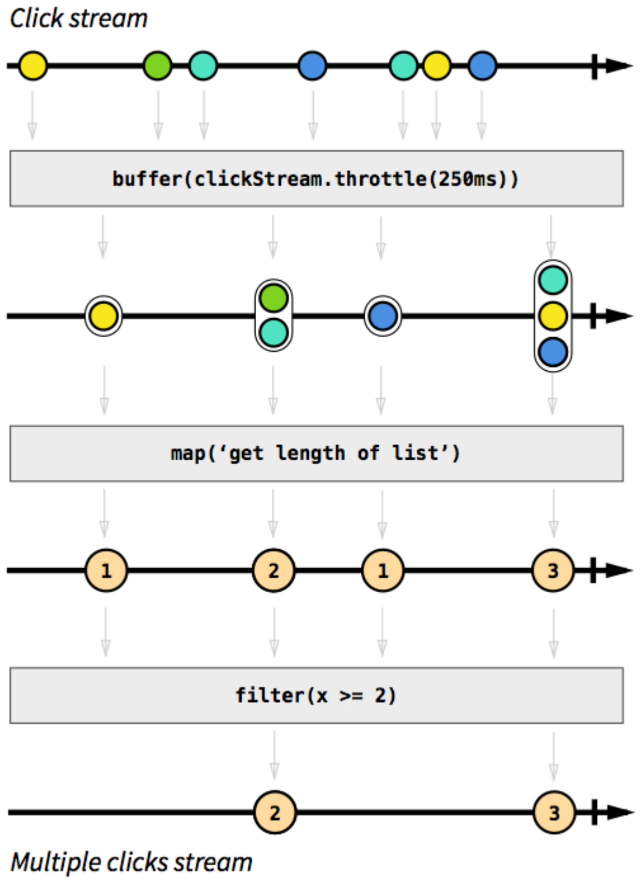


What is reactive?



Click stream
buffer(clickStream.throttle(250ms))
map('get length of list')
1
2
1
3

What is reactive?


Click stream
buffer(clickStream.throttle(250ms))
map('get length of list')
1
2
1
3
filter(x >= 2)
2
3
Multiple clicks stream

What is reactive?

Reactive programming is programming with asynchronous data streams

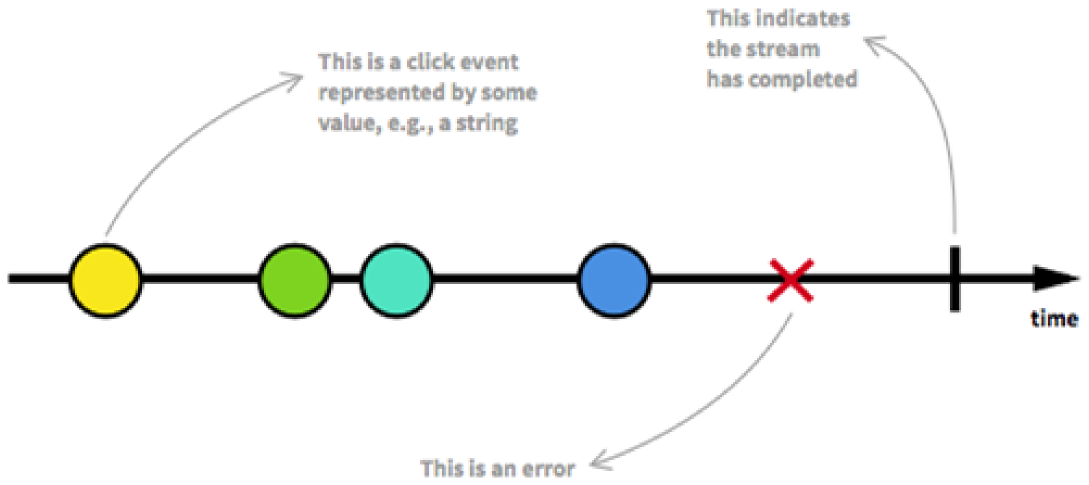
- What is reactive programming?

- Concepts

- Let’s Code!

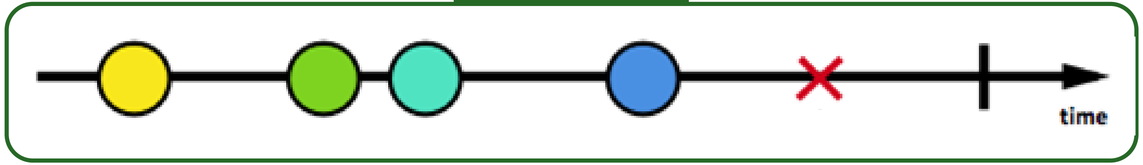
Concepts

The Observable


This is a click event represented by some value, e.g., a string
This indicates the stream has completed
time
This is an error

Concepts

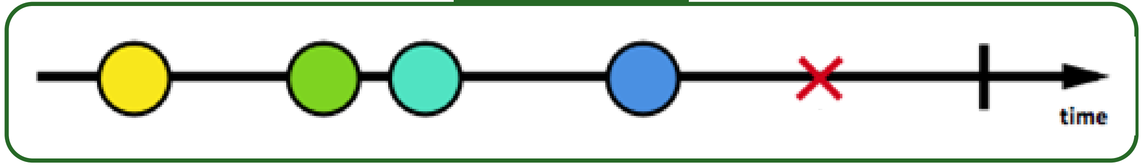
Observable



futurice LEAN SERVICE CREATION

Concepts

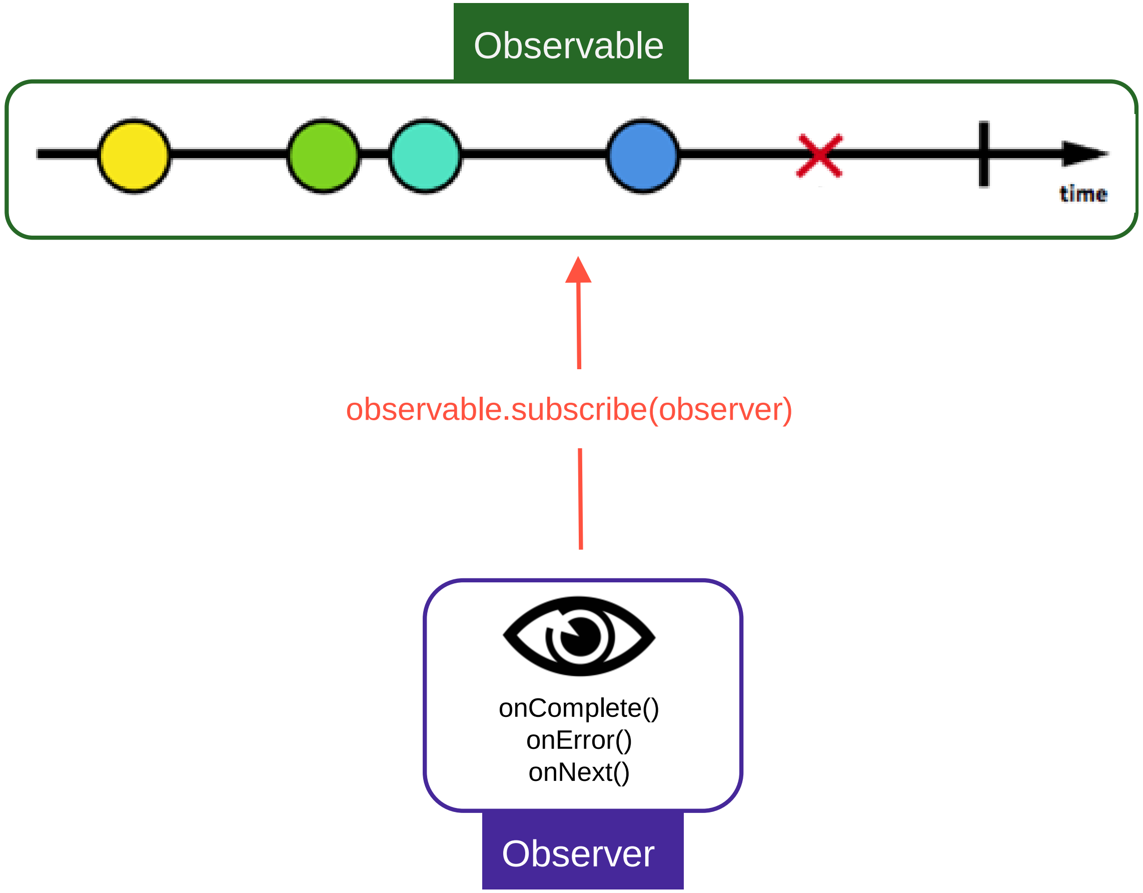
Observable



onComplete()
onError()
onNext()

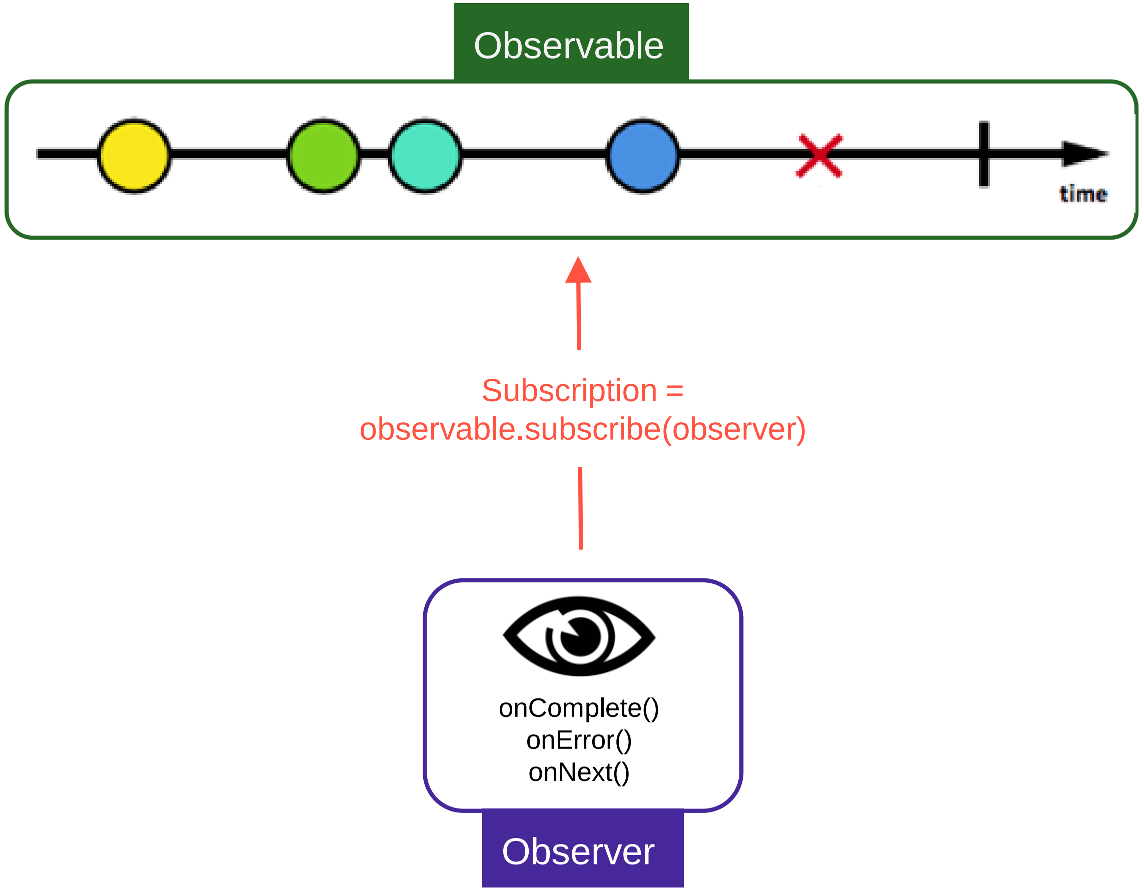
Observer

Concepts



futurice LEAN SERVICE CREATION

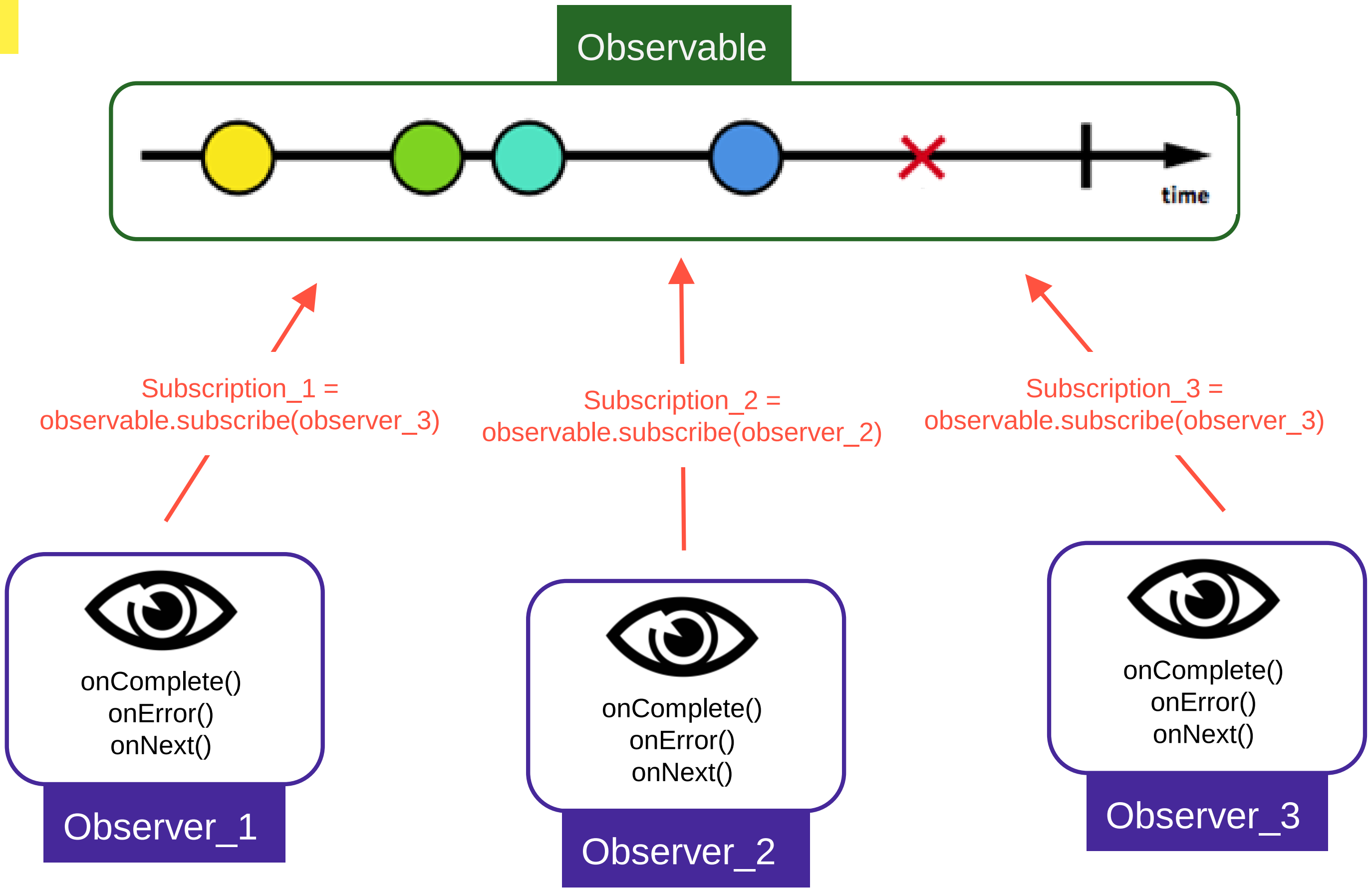
Concepts



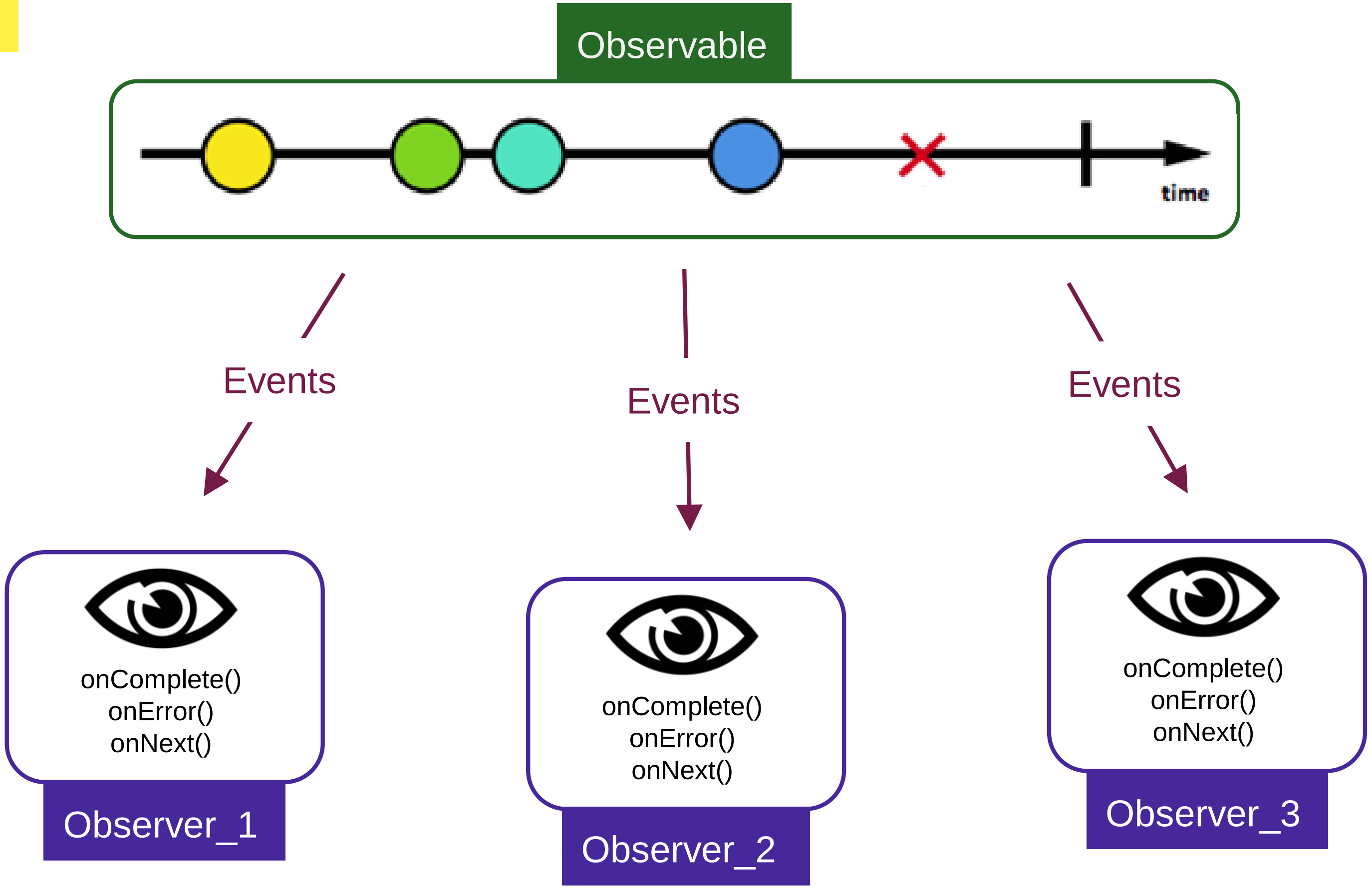
Subscription = observable.subscribe(observer)

onComplete()
onError()
onNext()

Concepts


Observable
time
Subscription_1 =
observable.subscribe(observer_3)
Subscription_2 =
observable.subscribe(observer_2)
Subscription_3 =
observable.subscribe(observer_3)
onComplete()
onError()
onNext()
Observer_1
onComplete()
onError()
onNext()
Observer_2
onComplete()
onError()
onNext()
Observer_3

Concepts


Observable
time
Events
Events
Events
onComplete()
onError()
onNext()
Observer_1
onComplete()
onError()
onNext()
Observer_2
onComplete()
onError()
onNext()
Observer_3

Concepts

- Observable = Stream, Producer
- Observer = Consumer
- Subscription = Contract

- What is reactive programming?
- Concepts
- Let’s Code!

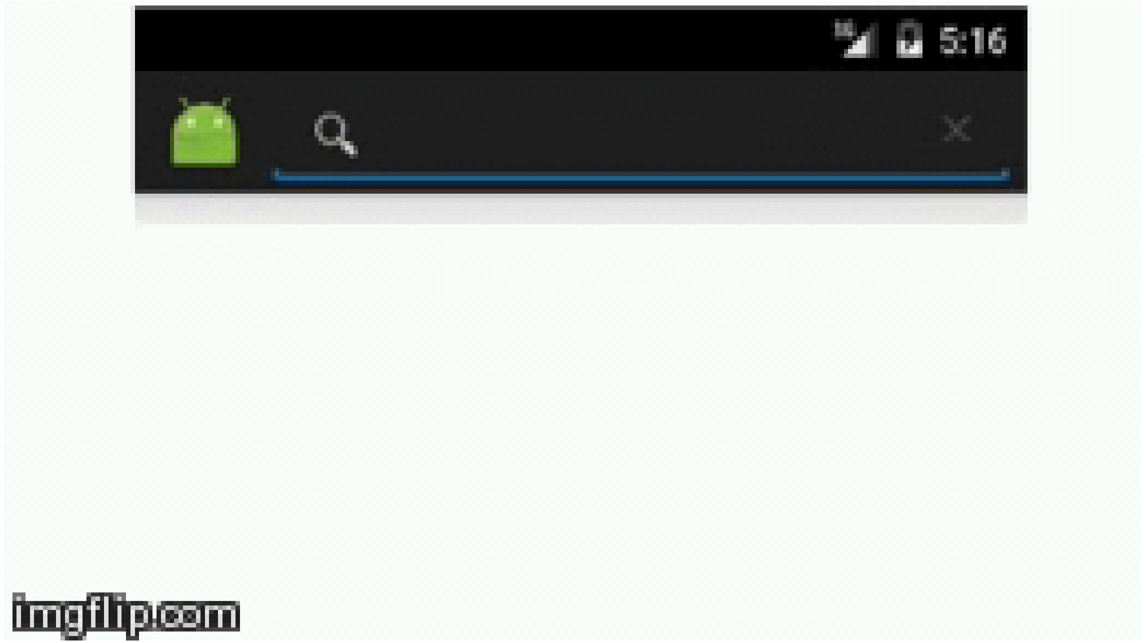
Let’s Code!

Search Widget Example

- Implement dynamic “search”

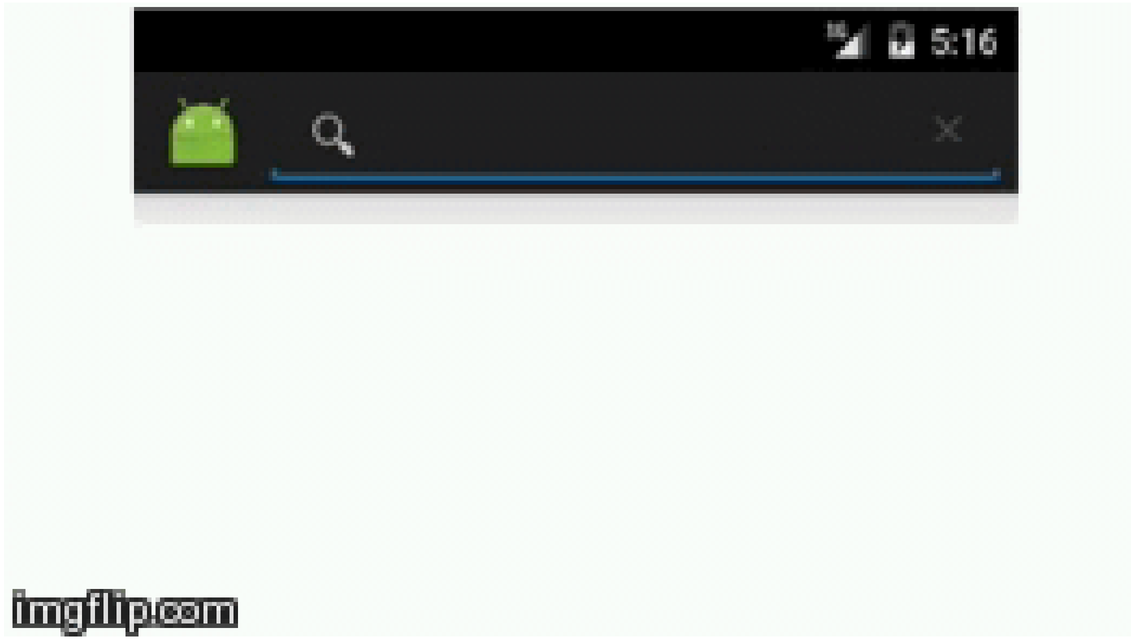
Let’s Code!

DON’T want to trigger a search for every letter



Let’s Code!

Trigger only after a small period of inactivity



Let's Code!

Search Widget Example

- Implement dynamic "search"
- Encrypt the query

futurice LEAN SERVICE CREATION

Let’s Code!

Search Widget Example

- Implement dynamic “search”
- Encrypt the query
- Do something if the query matches the key word

futurice LEAN SERVICE CREATION

Let’s Code!

RxJava by Square

Let’s Code!

Subject = Observable + Observer

Let’s Code!

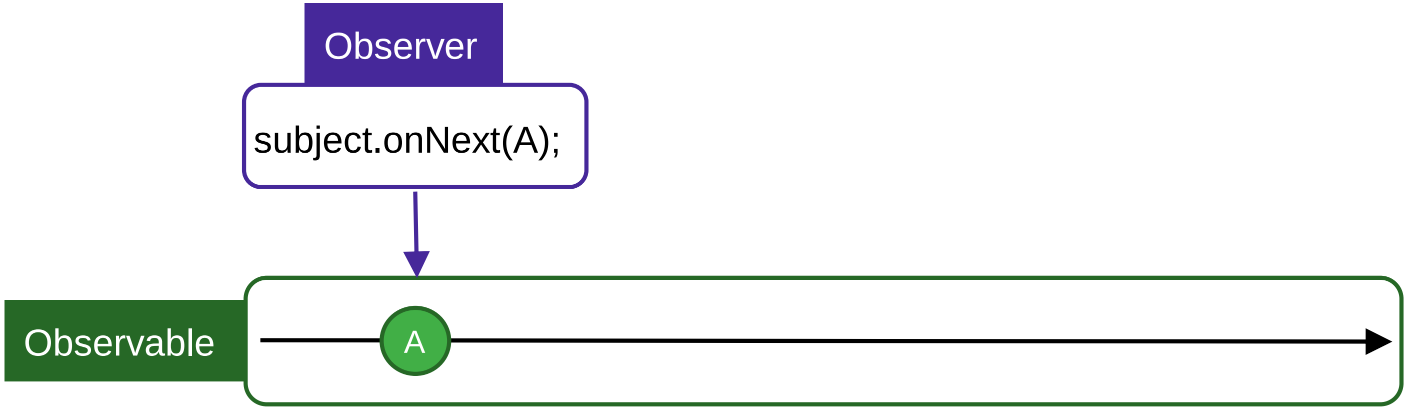
Subject = Observable + Observer

Observer

```
subject.onNext(A);
```

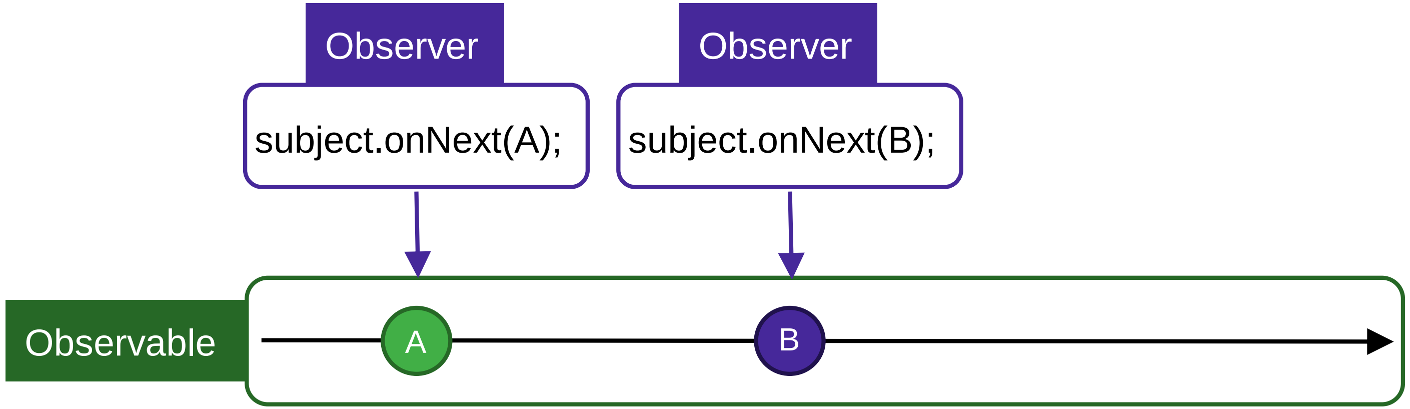
Let’s Code!

Subject = Observable + Observer



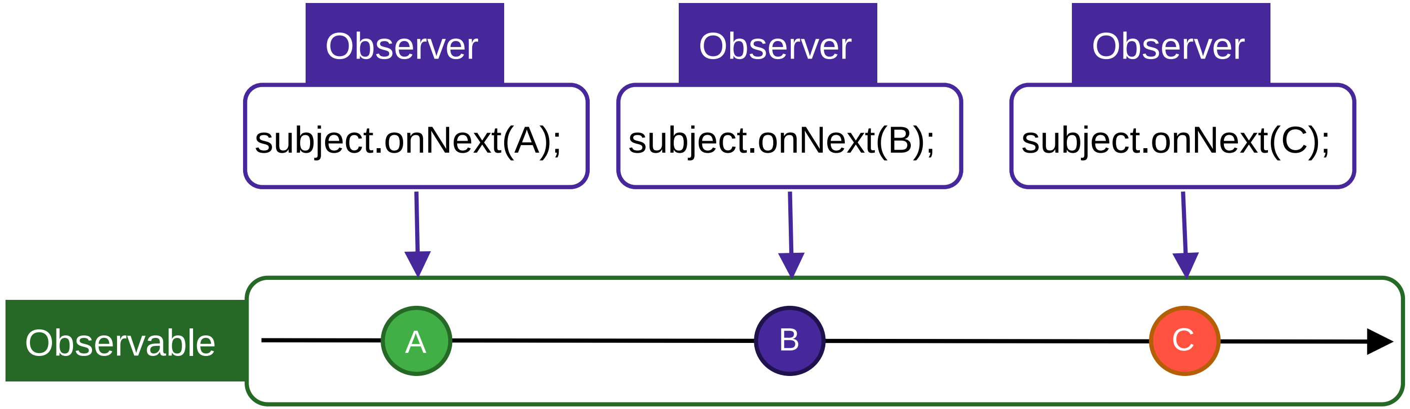
Let's Code!

Subject = Observable + Observer

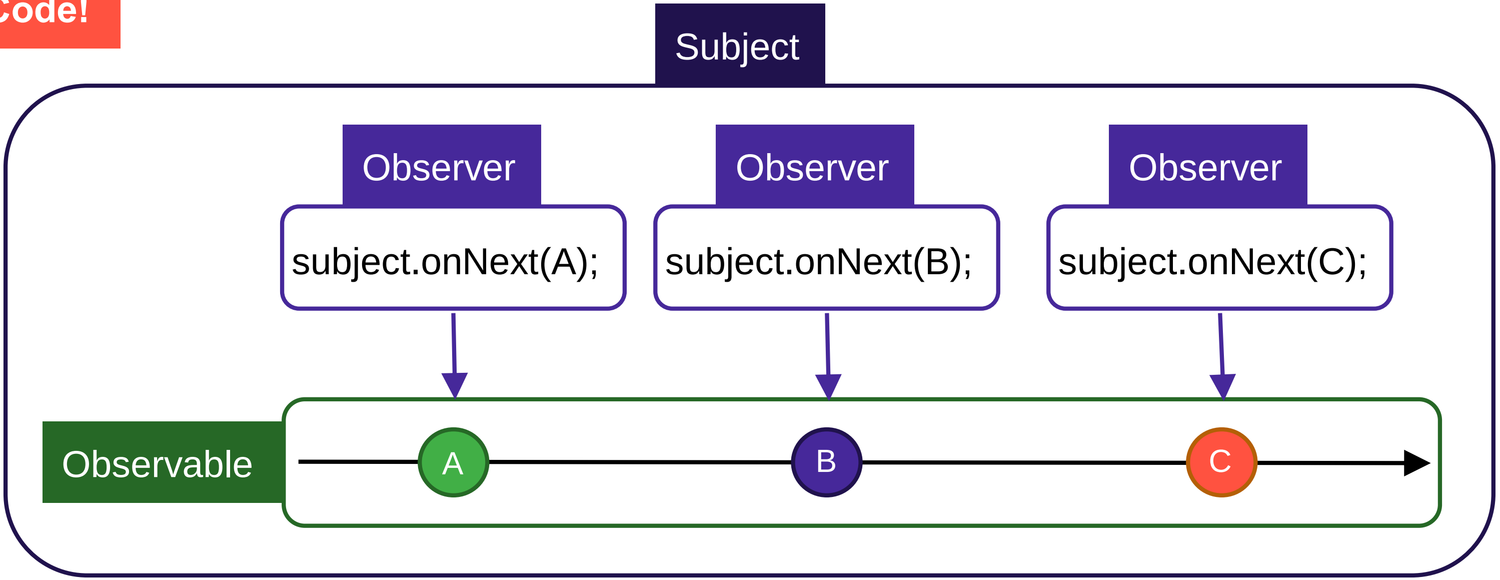


Let's Code!

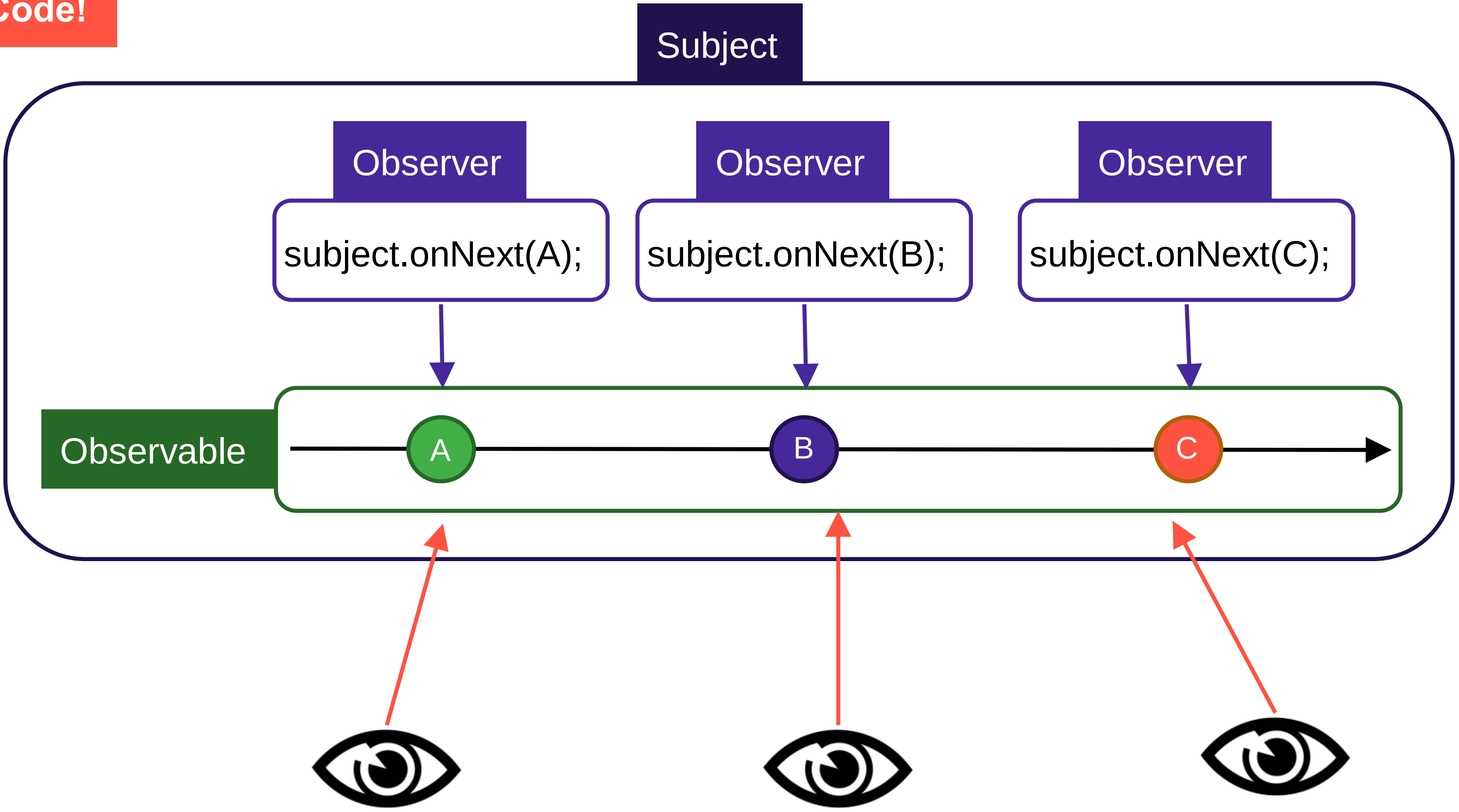
Subject = Observable + Observer



Let's Code!


Subject
Observer
subject.onNext(A);
Observer
subject.onNext(B);
Observer
subject.onNext(C);
Observable
A
B
C

Let's Code!



Subject
Observer
Observer
Observer
subject.onNext(A);
subject.onNext(B);
subject.onNext(C);
Observable
A
B
C

Reactive programming changes the way you think

WELCOME

TO THE DARK SIDE

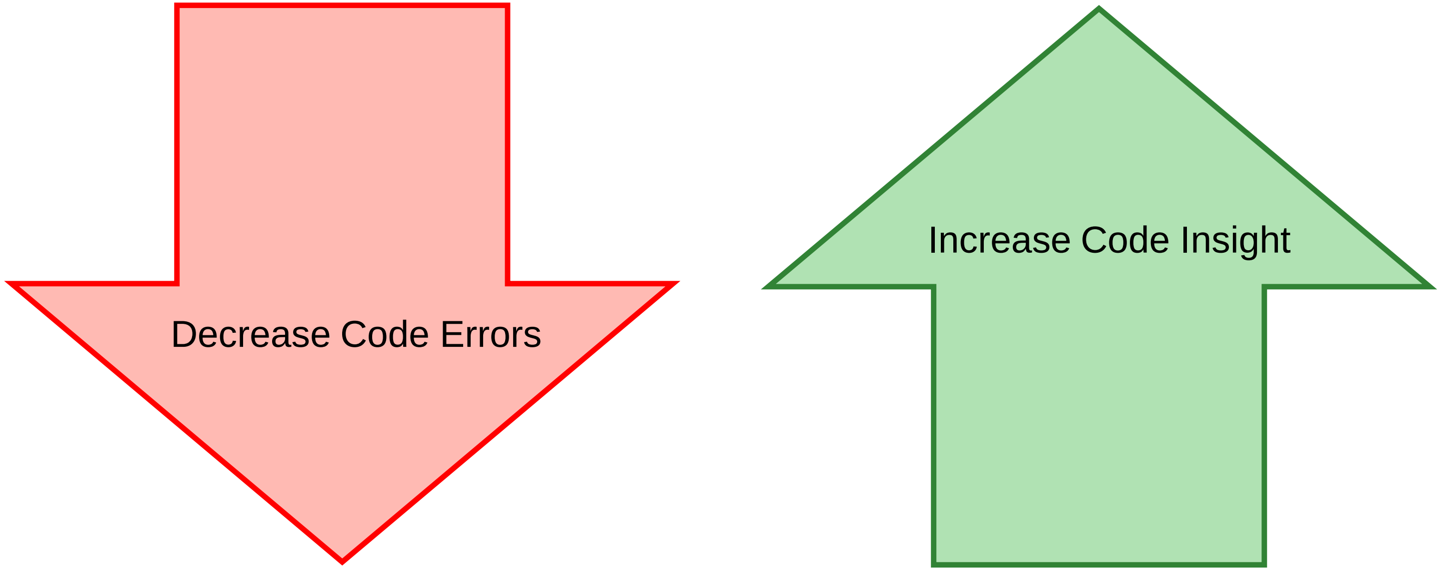
Reactive Cascade

Functional, Reactive, Different

“How do we develop easily
if everything was concurrent
by default?”

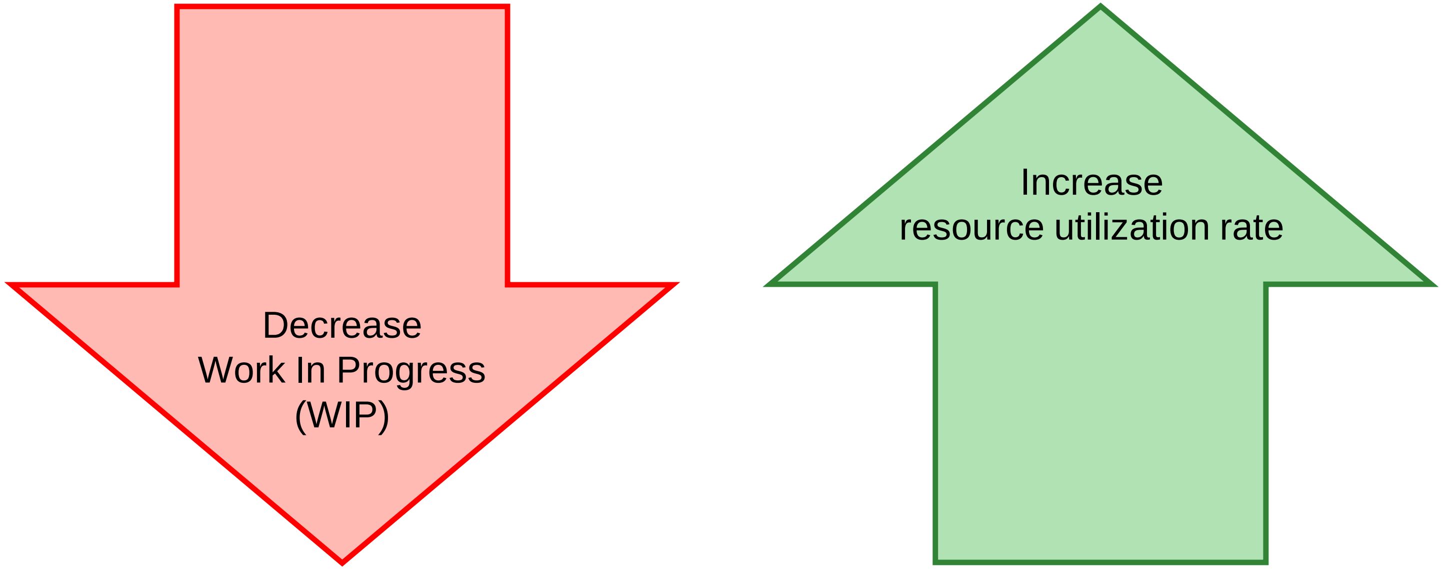
Psychology

How to increase developer performance



Information Logistics

How to increase runtime performance



The Driver: Great UX

- (designer blah blah here)
- Developer: **architecture details**
  - **Task throughput**
    - **Speed over time**
  - **Task responsiveness**
    - **UX latency, first things first**
  - **Reliability**
    - **Trust contract with the user**

https://www.flickr.com/photos/wonderlane/7167097899/in/photostream/lightbox/

Functional Example

```
ImmutableValue<Integer> count = new ImmutableValue<>();

// I don’t yet know the value
// But I do know what I want to do when the value is determined
count.then(value -> println(“The count is “ + value));
..
count.set(34);
// Triggers one time logic run
// Count is now an immutable value object
// Clean for further functional use
```

The point: safely and easily throw logic around – it can happen concurrently on any thread

Cascade

Functional Example

```
SettableAltFuture<Integer> count = new SettableAltFuture<>(WORKER);
count.then(value -> recalculateSheetOne(value); // CORE 1
count.then(value -> recalculateSheetTwo(value); // CORE 2
count.then(UI, value -> println("The count is " + value)) // CORE 3
..
count.set(34); // ANY THREAD
```

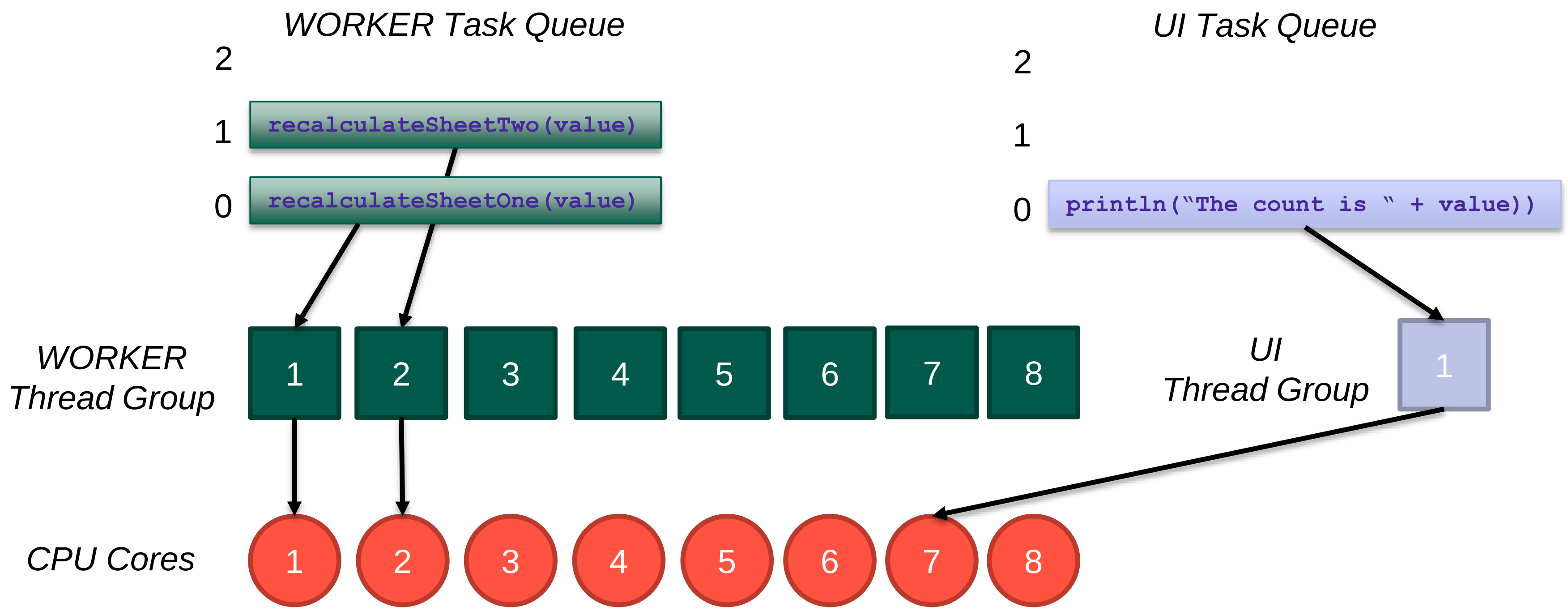
Atomic operation from any thread triggers transform to immutable value object

Three down-chain actions run concurrently on the UI and two worker threads

futurice LEAN SERVICE CREATION

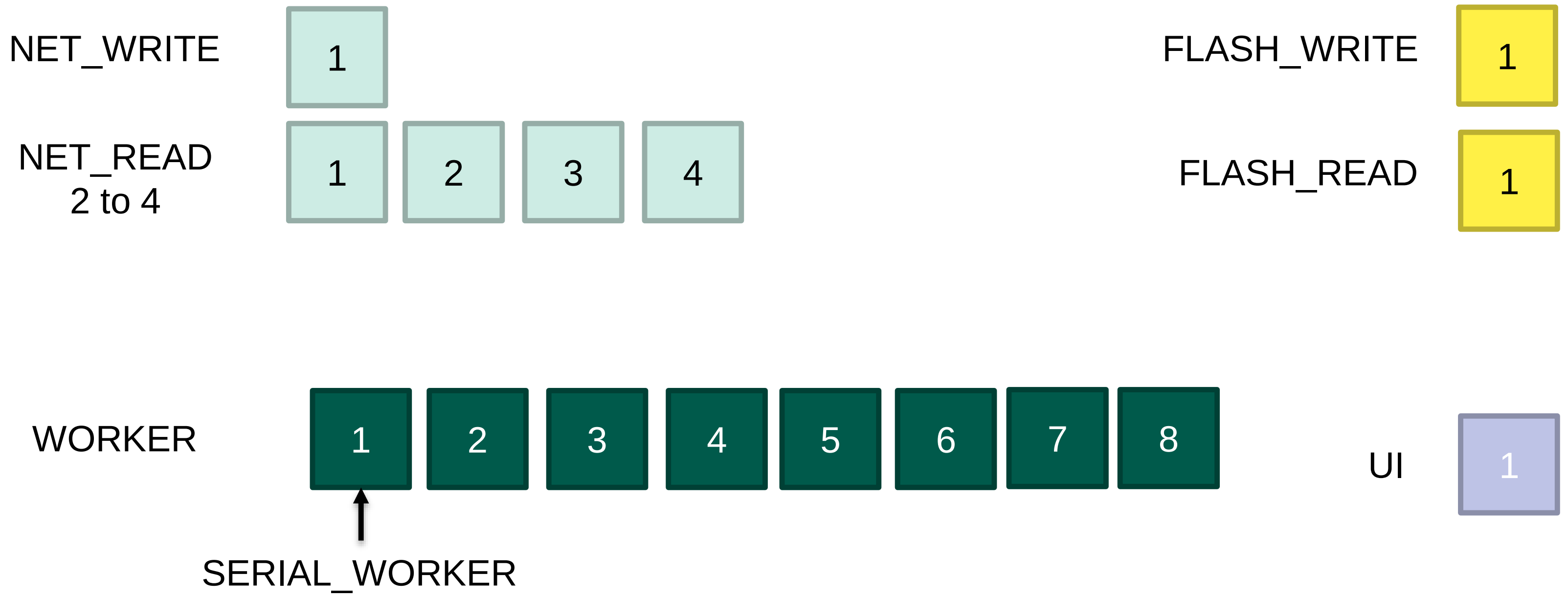
Cascade

Resource constraints determined execution thread groups:
WORKER, UI, NET_READ, NET_WRITE, FLASH_READ, FLASH_WRITE



Cascade

Pre-made System Resource Constraint Thread Groups (or add your own)



Cascade

Functional and Reactive: Use What Makes Sense To You

Functional

`.then(action)`

Fires once with `.fork()`

-> Great for adapting existing libraries

Reactive

`.subscribe(action)`

Fires each time the source changes

Automatic cleanup

-> Great for MVVM
-> Great for fast business logic

Reactive Concurrent Example

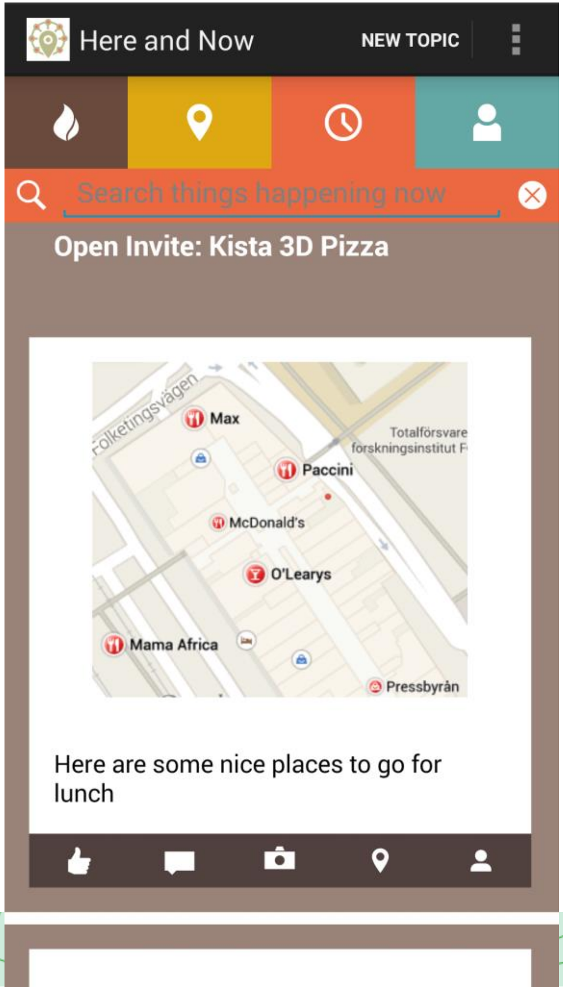
```
AtomicValue<Integer> angle = new AtomicValue<>(WORKER, “Angle”);

angle.subscribe(degrees -> viewModel.updateAngle(degrees))
     .subscribe(degrees -> mapToRadians(degrees))
     .subscribe(UI, radians -> updateDialOnScreen(radians));

..

angle.set(90); // From any thread
// -> Cascade of concurrent downchain actions on each set
```

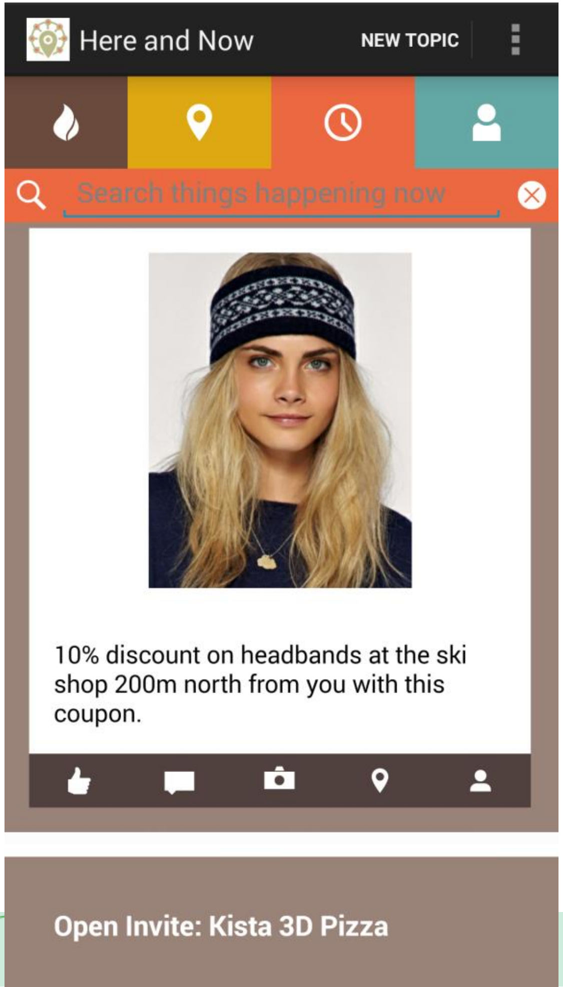
Cascade


Here and Now NEW TOPIC

Search things happening now

Open Invite: Kista 3D Pizza

Here are some nice places to go for lunch


Here and Now NEW TOPIC

Search things happening now

10% discount on headbands at the ski shop 200m north from you with this coupon.

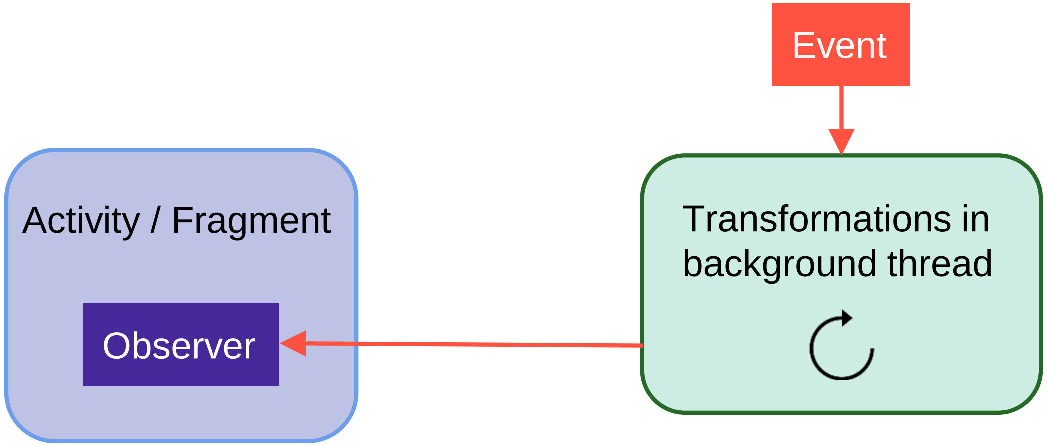
Open Invite: Kista 3D Pizza

The issues

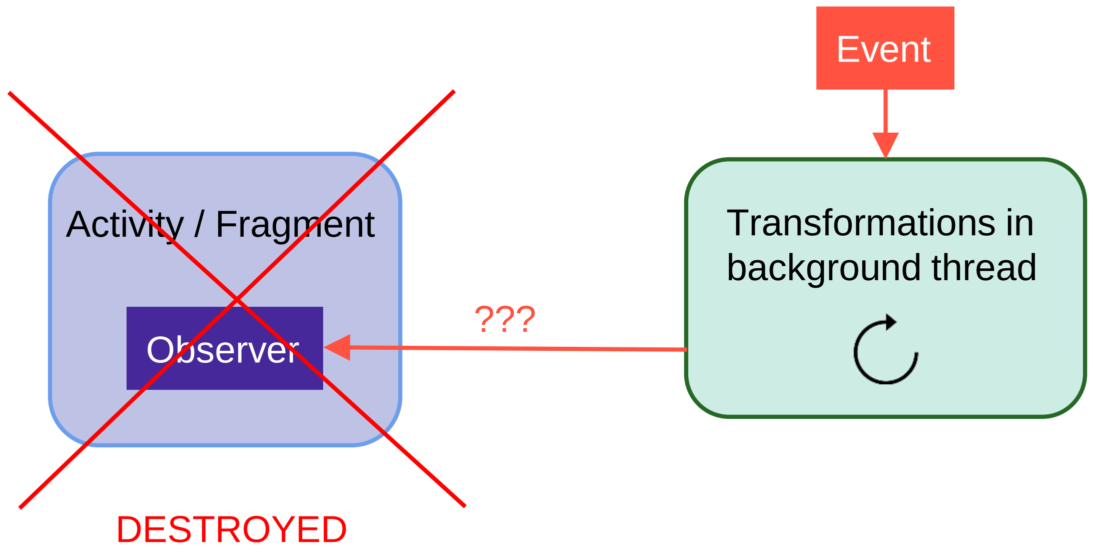
- Activity/Fragment life-cycle
- Error handling
- Logs
- Back Pressure
- Hot & Cold Observables

Life-cycle

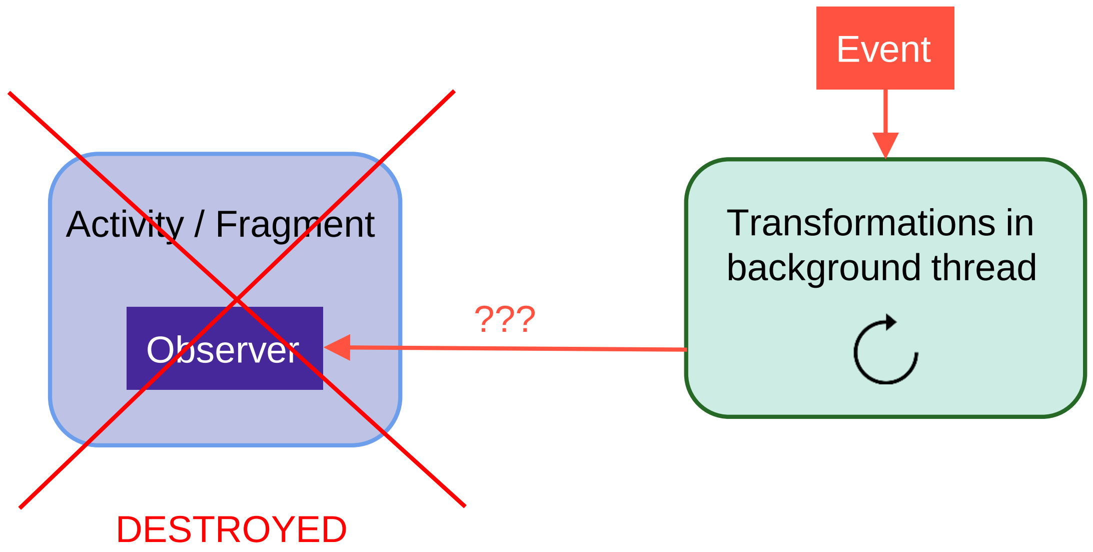
RxJava



Life-cycle


Event
Transformations in background thread
Activity / Fragment
Observer
???
DESTROYED

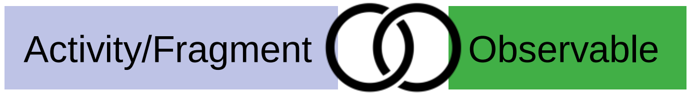
Life-cycle



EXCEPTION + MEMORY LEAK

Life-cycle

Bind the observable to the Activity/Fragment life-cycle



```
AppObservable.bindActivity(activity, observable);
AppObservable.bindFragment(fragment, observable);
```

Handle the subscriptions & unsubscribe!

Enable Strict Mode

Cleanup, unsubscribe, boring plumbing work prone to errors..

```
// This space intentionally left blank
```

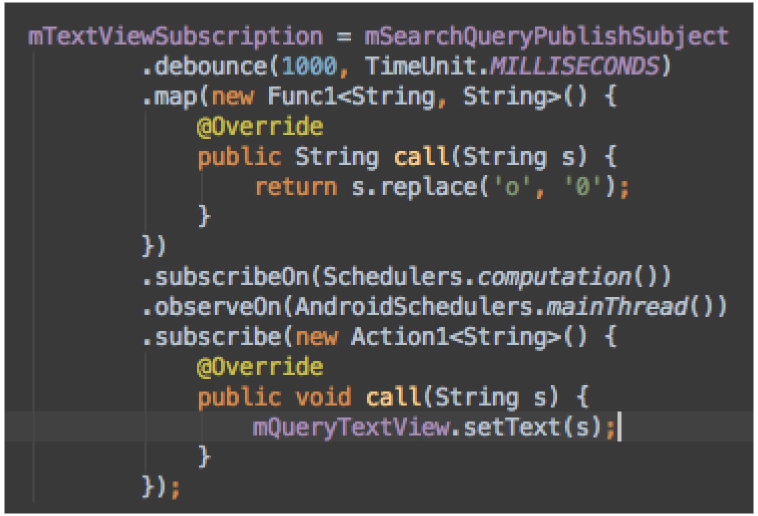
Pl

Error Handling

RxJava

futurice LEAN SERVICE CREATION

Actions



```
mTextViewSubscription = mSearchQueryPublishSubject
        .debounce(1000, TimeUnit.MILLISECONDS)
        .map(new Func1<String, String>() {
            @Override
            public String call(String s) {
                return s.replace('o', '0');
            }
        })
        .subscribeOn(Schedulers.computation())
        .observeOn(AndroidSchedulers.mainThread())
        .subscribe(new Action1<String>() {
            @Override
            public void call(String s) {
                mQueryTextView.setText(s);
            }
        });
```

Error Handling

Error event ends the subscription

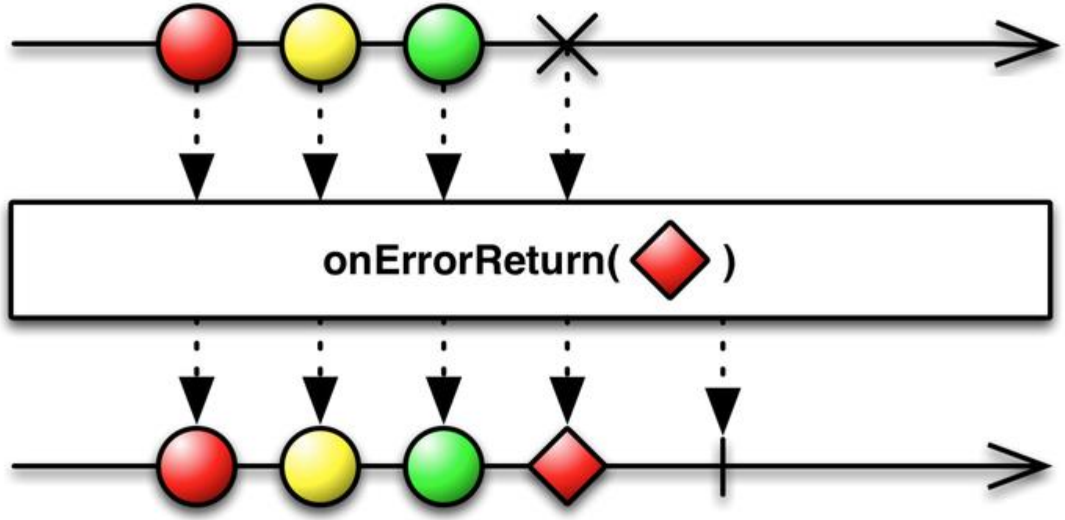
Error Handling

Error event ends the subscription

There are some workarounds $\longrightarrow$ Operators

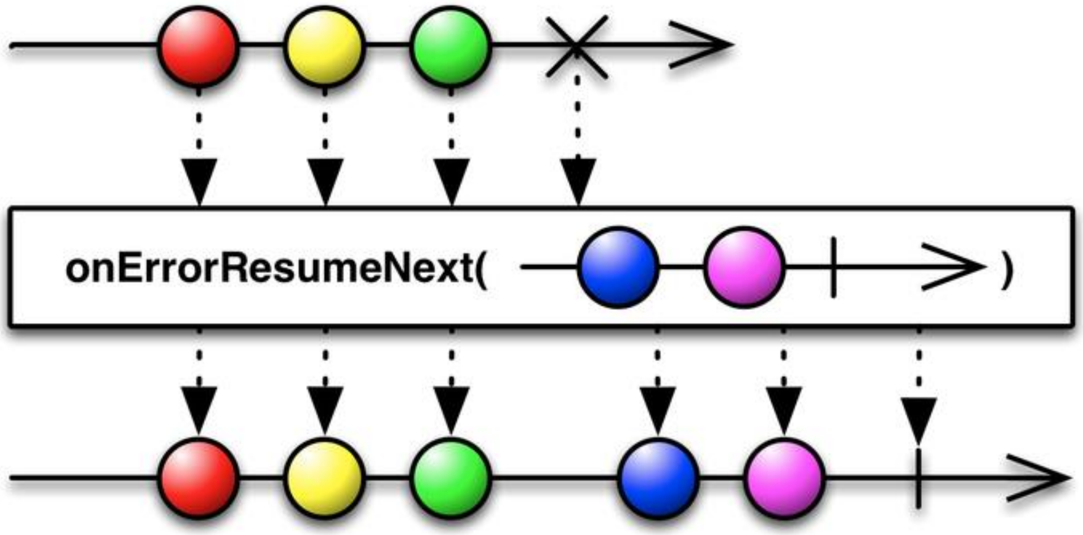
Error Handling

Operators for Error Handling



Error Handling

Operators for Error Handling



Error Handling

Cascade

futurice LEAN SERVICE CREATION

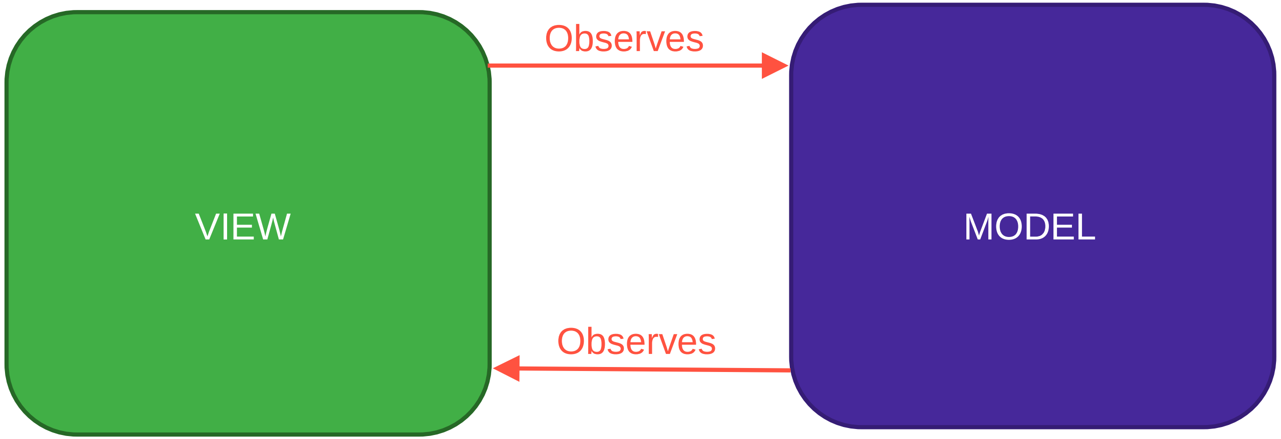
Error Handling

```
NET_READ.then(() ->
                return getMessagesServer())
      .then(WORKER, (raw) -> {
                return parseMessages(raw))
      }
      .then(NET_WRITE, (MyDataType parsedList) -> {
                storeToFlash(parsedList);
      }
      .then(UI, (MyDataType parsedList) -> {
                display(result))
      }
      .onError(() ->
                popupError("Stopped at count " + count.get());
```

Error handling is automatic and fail-fast in debug builds

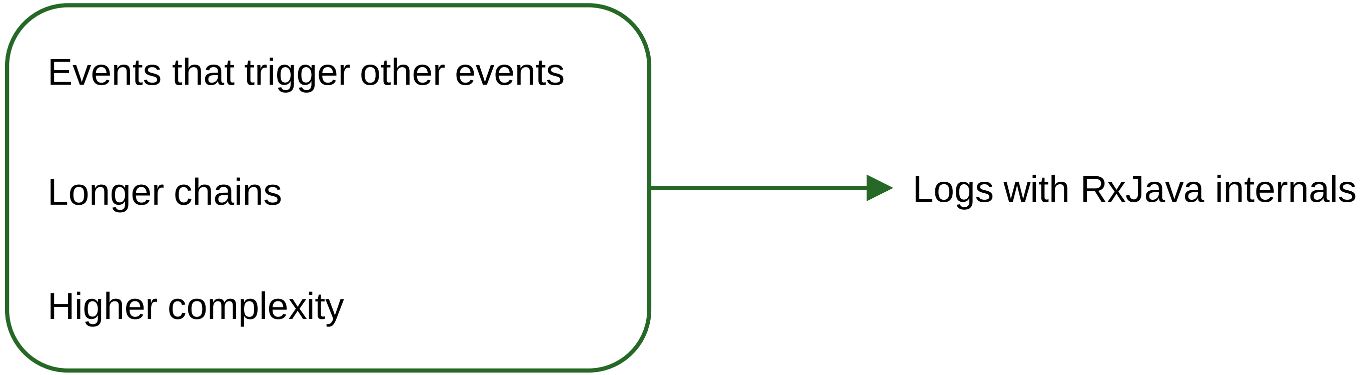
You can add your own in-chain actions

Using RxJava everywhere



Logs

Where is the trigger?

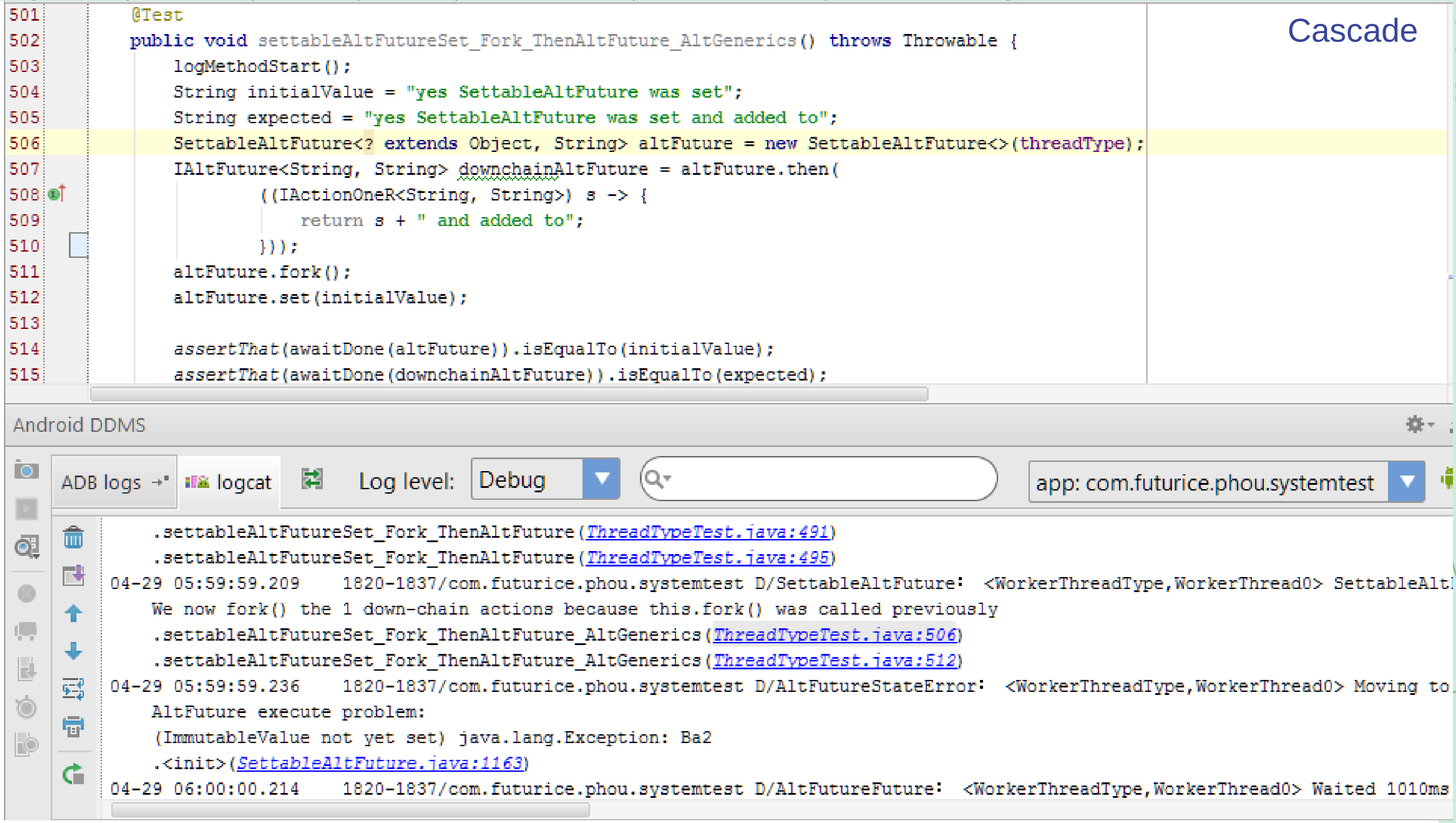


DIFFICULT TO TRACK DOWN!

Logs

Cascade

futurice LEAN SERVICE CREATION



```
501     @Test
502     public void settableAltFutureSet_Fork_ThenAltFuture_AltGenerics() throws Throwable {
503         logMethodStart();
504         String initialValue = "yes SettableAltFuture was set";
505         String expected = "yes SettableAltFuture was set and added to";
506         SettableAltFuture<? extends Object, String> altFuture = new SettableAltFuture<>(threadType);
507         IAltFuture<String, String> downchainAltFuture = altFuture.then(
508                 ((IActionOneR<String, String>) s -> {
509                     return s + " and added to";
510                 }));
511         altFuture.fork();
512         altFuture.set(initialValue);
513 
514         assertThat(awaitDone(altFuture)).isEqualTo(initialValue);
515         assertThat(awaitDone(downchainAltFuture)).isEqualTo(expected);
```

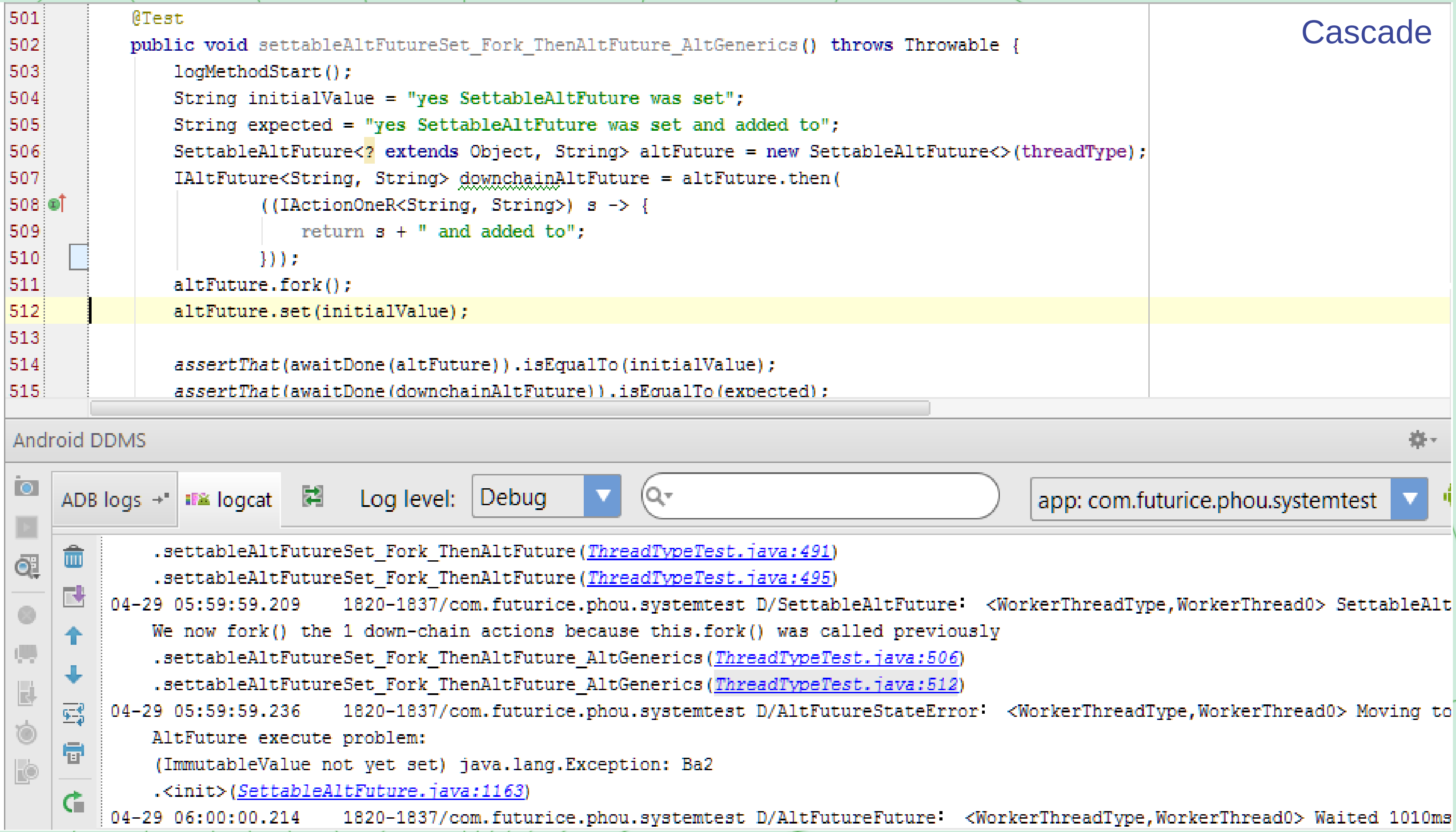
Android DDMS

ADB logs | logcat | Log level: Debug | app: com.futurice.phou.systemtest

```
    .settableAltFutureSet_Fork_ThenAltFuture(ThreadTypeTest.java:491)
    .settableAltFutureSet_Fork_ThenAltFuture(ThreadTypeTest.java:495)
04-29 05:59:59.209    1820-1837/com.futurice.phou.systemtest D/SettableAltFuture:  <WorkerThreadType,WorkerThread0> SettableAlt
    We now fork() the 1 down-chain actions because this.fork() was called previously
    .settableAltFutureSet_Fork_ThenAltFuture_AltGenerics(ThreadTypeTest.java:506)
    .settableAltFutureSet_Fork_ThenAltFuture_AltGenerics(ThreadTypeTest.java:512)
04-29 05:59:59.236    1820-1837/com.futurice.phou.systemtest D/AltFutureStateError:  <WorkerThreadType,WorkerThread0> Moving to
    AltFuture execute problem:
    (ImmutableValue not yet set) java.lang.Exception: Ba2
    .<init>(SettableAltFuture.java:1163)
04-29 06:00:00.214    1820-1837/com.futurice.phou.systemtest D/AltFutureFuture:  <WorkerThreadType,WorkerThread0> Waited 1010ms
```

Logs

Cascade



```
501     @Test
502     public void settableAltFutureSet_Fork_ThenAltFuture_AltGenerics() throws Throwable {
503         logMethodStart();
504         String initialValue = "yes SettableAltFuture was set";
505         String expected = "yes SettableAltFuture was set and added to";
506         SettableAltFuture<? extends Object, String> altFuture = new SettableAltFuture<>(threadType);
507         IAltFuture<String, String> downchainAltFuture = altFuture.then(
508                 ((IActionOneR<String, String>) s -> {
509                     return s + " and added to";
510                 }));
511         altFuture.fork();
512         altFuture.set(initialValue);
513
514         assertThat(awaitDone(altFuture)).isEqualTo(initialValue);
515         assertThat(awaitDone(downchainAltFuture)).isEqualTo(expected);
```

```
    .settableAltFutureSet_Fork_ThenAltFuture(ThreadTypeTest.java:491)
    .settableAltFutureSet_Fork_ThenAltFuture(ThreadTypeTest.java:495)
04-29 05:59:59.209    1820-1837/com.futurice.phou.systemtest D/SettableAltFuture:  <WorkerThreadType,WorkerThread0> SettableAlt
    We now fork() the 1 down-chain actions because this.fork() was called previously
    .settableAltFutureSet_Fork_ThenAltFuture_AltGenerics(ThreadTypeTest.java:506)
    .settableAltFutureSet_Fork_ThenAltFuture_AltGenerics(ThreadTypeTest.java:512)
04-29 05:59:59.236    1820-1837/com.futurice.phou.systemtest D/AltFutureStateError:  <WorkerThreadType,WorkerThread0> Moving to
    AltFuture execute problem:
    (ImmutableValue not yet set) java.lang.Exception: Ba2
    .<init>(SettableAltFuture.java:1163)
04-29 06:00:00.214    1820-1837/com.futurice.phou.systemtest D/AltFutureFuture:  <WorkerThreadType,WorkerThread0> Waited 1010ms
```

Back Pressure

What if the observable produces faster than the observer can consume?

Back Pressure

What if the observable produces faster than the observer can consume?

It has to go through all the transformations $\longrightarrow$ Takes time

Back Pressure

What if the observable produces faster than the observer can consume?

It has to go through all the transformations $\longrightarrow$ Takes time

How to handle backpressure:
https://github.com/ReactiveX/RxJava/wiki/Backpressure

Back Pressure

The Good

- We use queues instead
- Concurrent, 4x or 8x faster
    - Generally no bogging
- Depth-first task tree execution
    - Low task latency once the chain starts
    - Low Work In Process (WIP)
- Reactive chains <u>pull</u> inputs
    - They “skip intermediate steps”

The Ugly

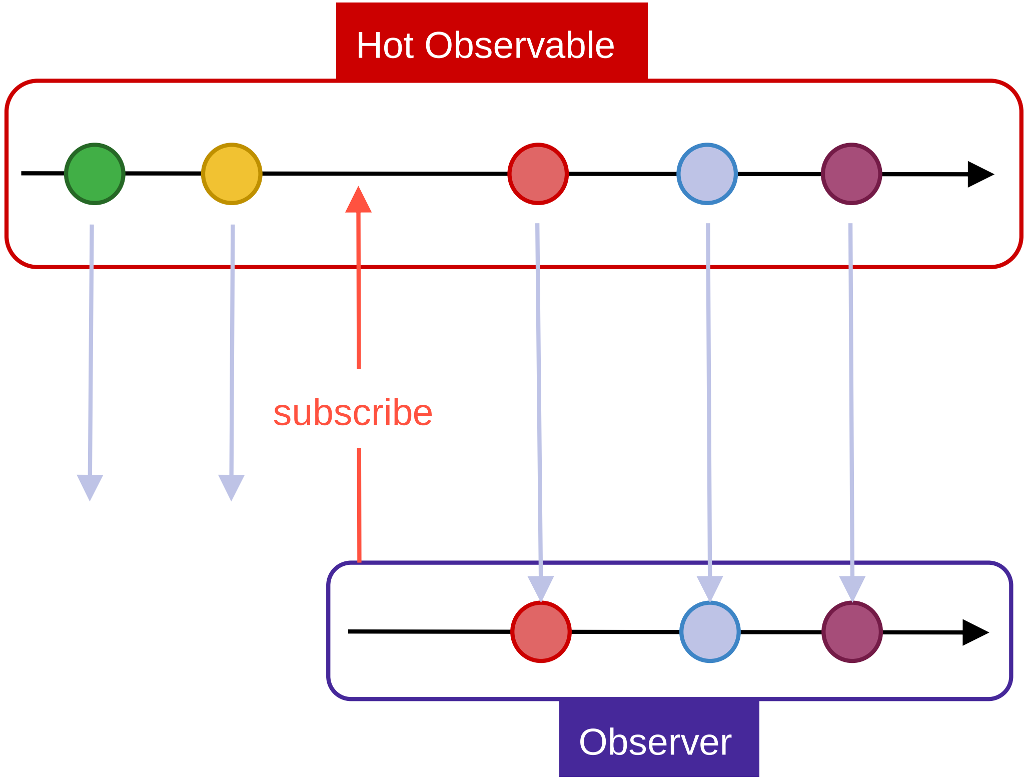
- Shared queue, shared problem
- The queue _can_ grow
    - Very slow tasks
    - Bad design
- Bypass a bogged executor
    - Create a dedicated executor
- Trigger source quench when the problem queue exceed a set length

Hot & Cold Observables

When does an Observable start emitting items?

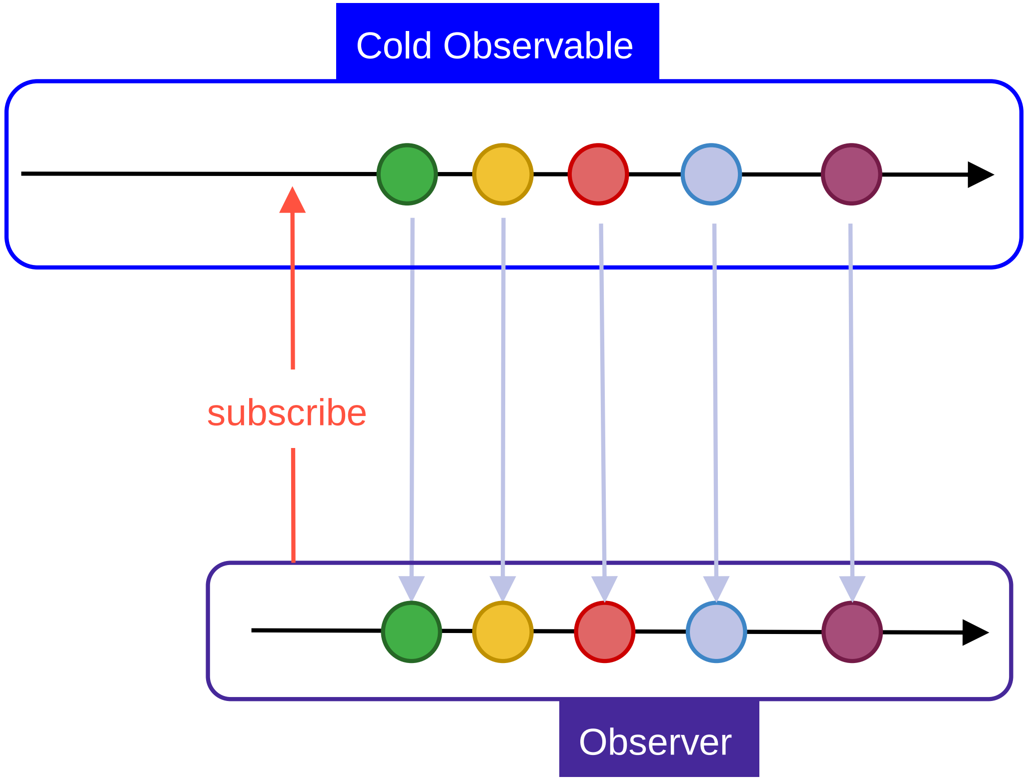
Hot & Cold Observables

RxJava



futurice LEAN SERVICE CREATION

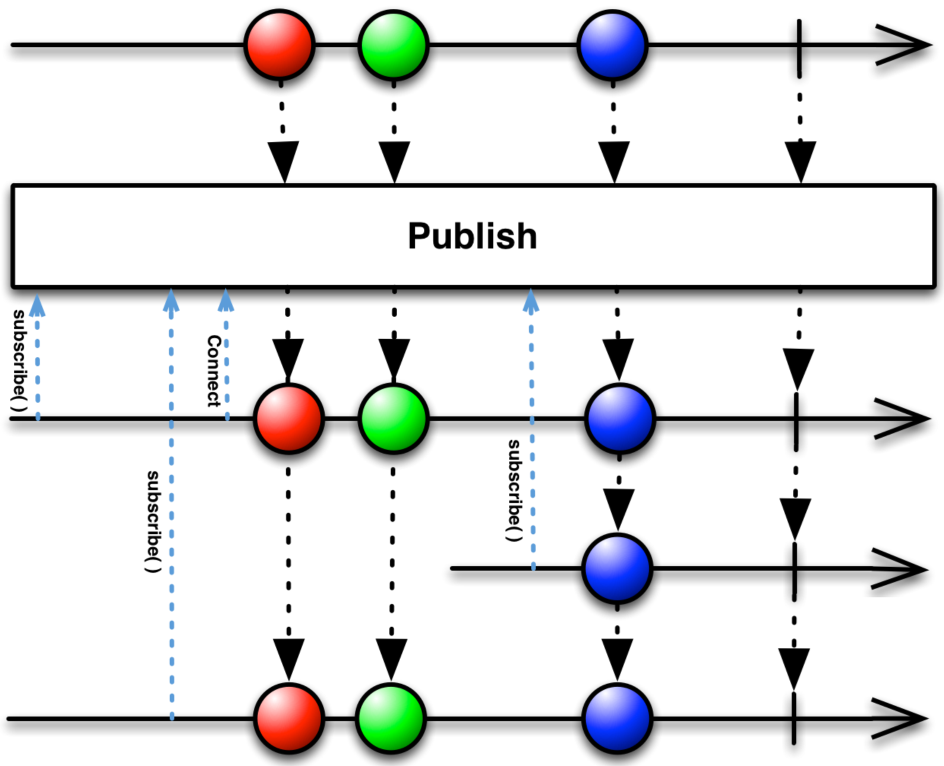
Hot & Cold Observables



Hot & Cold Observables

RxJava

Connectable Observable



futurice LEAN SERVICE CREATION

Hot & Cold Observables

Simple: Everything is cold

Functional
- **.fork()** the chain to queue for execution

Reactive
- **.fire()** to force current value to (re)evaluate the chain
- Or wait for the next **.set(value)** change

WELCOME

TO THE DARK SIDE

The introduction to RxJava you've been missing:
https://gist.github.com/staltz/868e7e9bc2a7b8c1f754

Reference architecture for Android using RxJava:
https://github.com/tehmou/rx-android-architecture

Don't break the chain:
http://blog.danlew.net/2015/03/02/dont-break-the-chain/

Reactive wiki:
http://reactivex.io/

Top 7 tips for RxJava on Android:
http://futurice.com/blog/top-7-tips-for-rxjava-on-android

Interactive diagrams of Rx Observables:
http://rxmarbles.com/

Reactive Cascade
https://github.com/paulirotta/cascade

LINK TO THIS PRESENTATION:

futurice LEAN SERVICE CREATION

Thanks!

lucia.payo@futurice.com

paul.houghton@futurice.com
@mobile_rat