RxJava cheat sheet

Basic RxJava classes

`Observable<T>` - emits 0 or n items and terminates with complete or an error.

`Single<T>` - emits either a single item or an error. The reactive version of a method call. You subscribe to a *Single* and you get either a return value or an error.

`Maybe<T>` - succeeds with either an item, no item, or errors. The reactive version of an *Optional*.

`Completable` - either completes or returns an error. It never return items. The reactive version of a *Runnable*.

Creating observables

Create an observable from a value, a collection or *iterable*, or a result of a *callable*:

```
Observable.just("RebelLabs");
Observable.fromIterable(iterable);
Observable.fromCallable(callable);
```

RxBindings

Turns Android UI events into RxJava observables:

```
Button button = (Button)
findViewById(R.id.button);
RxView.clicks(button).subscribe(x -> {
  // do work here
});
```

RxAndroid

Control on which threads you observe and react to events (avoid long computations on the main thread):

```
Observable.just("RebelLabs")
.subscribeOn(Schedulers.newThread())
.observeOn(AndroidSchedulers.mainThread())
.subscribe(anObserver);
```

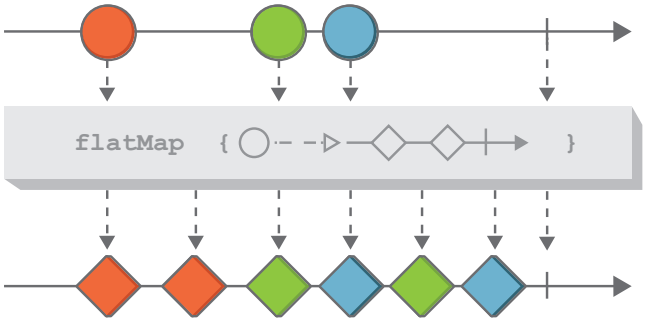
Data processing functions

`map(Function<? super T,? extends R> mapper)` - applies a function to each of items, and emits the returned values.

`filter(Predicate<? super T> predicate)` - emits only the items satisfying a predicate.

`buffer(int count)` - emits lists of the items of the specified size.

`zip(ObservableSource s1, ObservableSource s2, BiFunction<T1, T2, R> f)` - applies a function to the items from multiple observables and emits the returned value.



`flatMap(Function<? super T,? extends ObservableSource<? extends R>> mapper)` - takes a function from items to an *Observable*, emits the items of the resulting *Observables*

.

`groupBy(Function<? super T,? extends K> keySelector)` - emits items grouped by a specified key selector function.

`timeout(long timeout, TimeUnit timeUnit)` - emits items of the original *Observable*. If the next item isn't emitted within the specified timeout, a *TimeoutException* occurs.

Subscribing to observables

Observers provide a mechanism for receiving data and notifications from *Observables* using the following API:

`onNext(T t)` - provides the *Observer* with a new item to observe.

`onError(Throwable e)` - notifies the *Observer* that the Observable has experienced an error condition.

`onComplete()` - notifies the *Observer* that the *Observable* has finished sending push-based notifications.

RxLifecycle

Bind subscription lifecycle to Android components. Destroy subscriptions and avoid memory leaks on destroy / pause events.

```
myObservable.compose(
  RxLifecycleAndroid.bindActivity(lifecycle))
  .subscribe();
```

Testing observables

`TestSubscriber` - a subscriber that records events that you can make assertions upon.

`TestObserver` - an *Observer* that records events that you can make assertions upon.

```
TestSubscriber<Integer> ts =
Flowable.range(1, 5).test();
// assert properties
assertThat(
      ts.values()).hasSize(5));
```