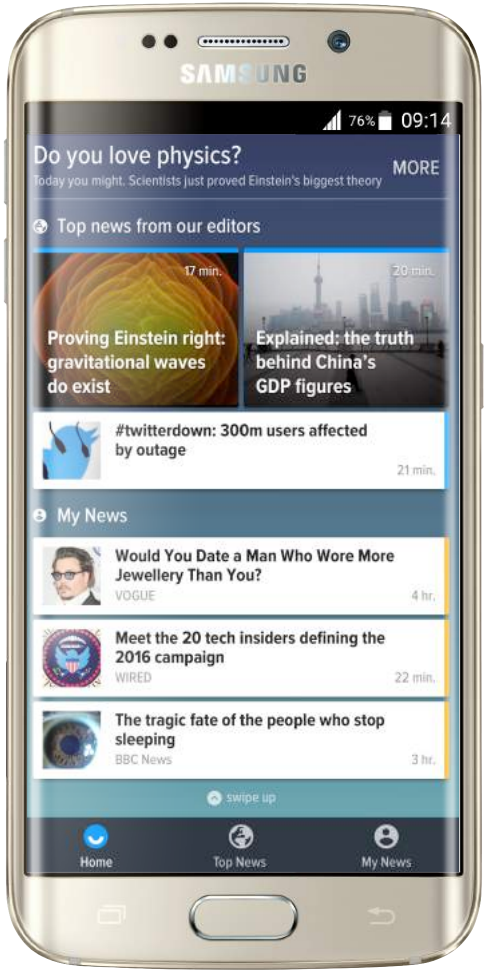
MVVM and RxJava – the perfect mix

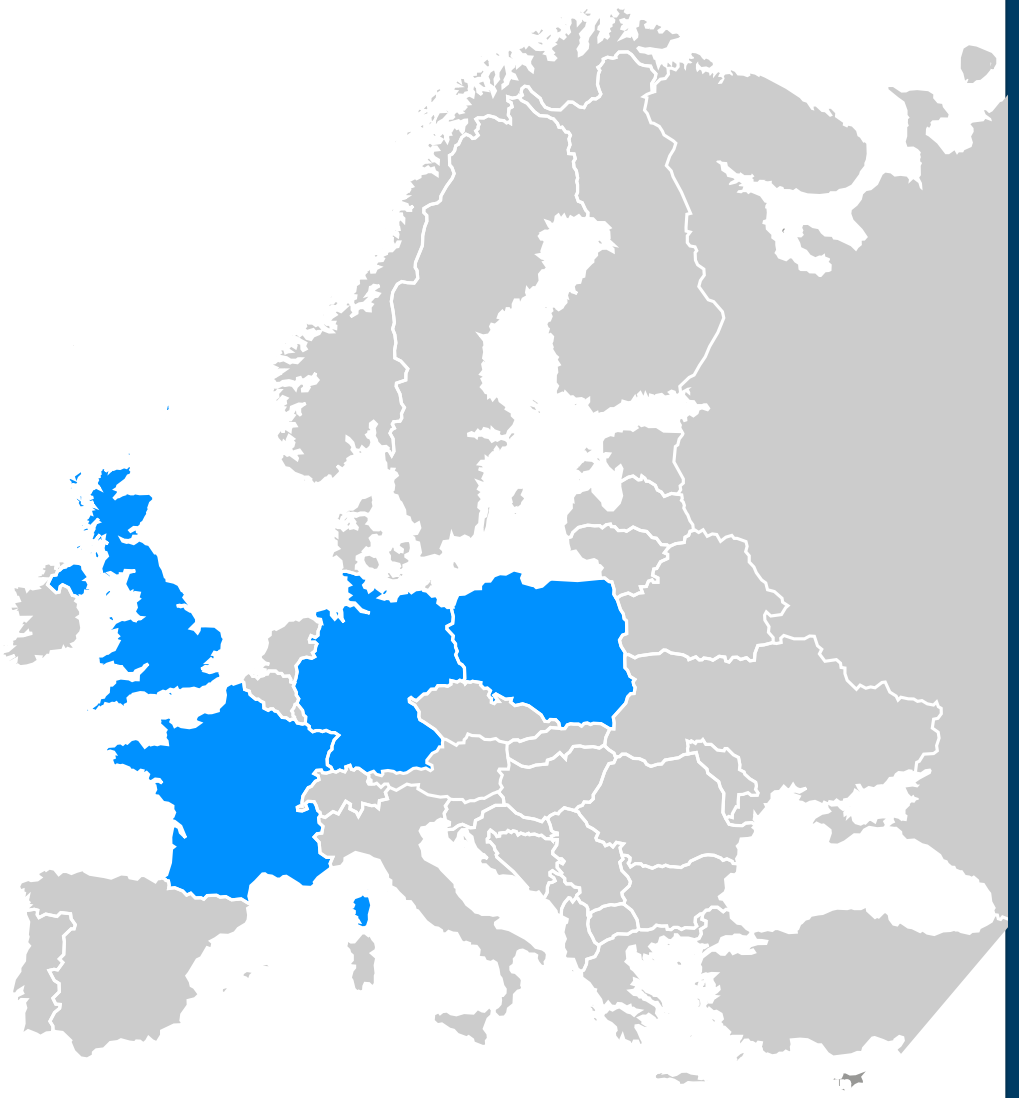
Droidcon Berlin

Florina Muntenescu
@FMuntenescu

upday

What and who upday is



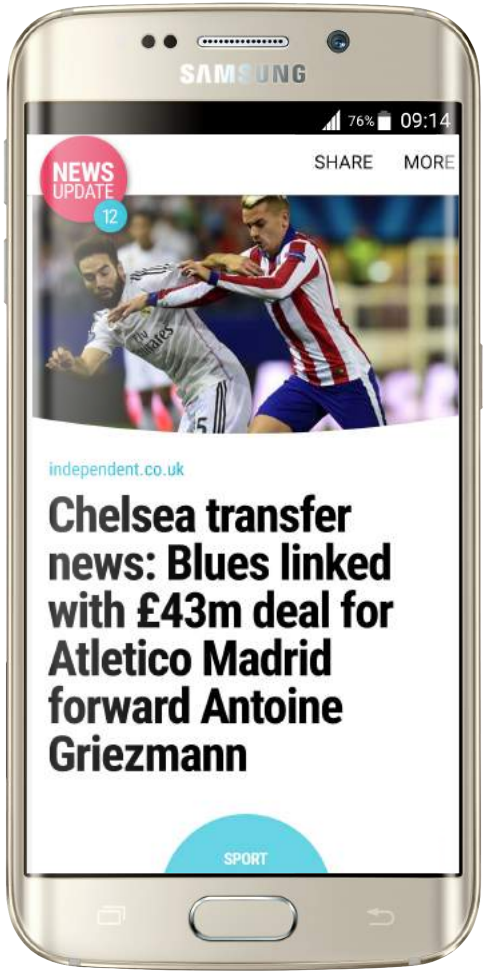


upday

When did we decide to go the RxJava & MVVM way?

upday

The old app



upday

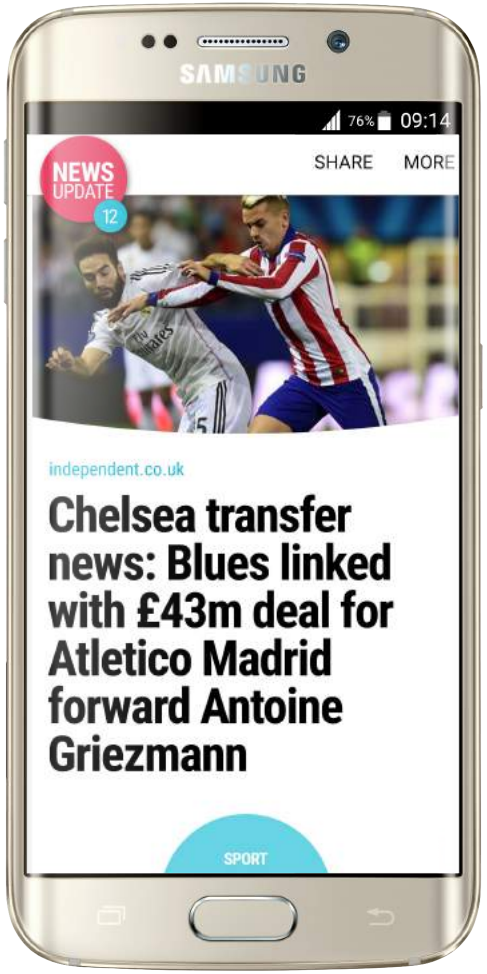
The old app

```
// TODO: Workaround for APP-124. Will be removed after refactoring.
if (mWtkSwipeLayout != null
    && mWtkSwipeLayout.getChildCount() > 1
    && isWtkEmptyVisible()
    && childCount == mWtkSwipeLayout.getMaxViewsCount()) {
    mWtkSwipeLayout.removeViewAt(1);
}
```

99 little bugs in the code.
99 little bugs in the code.
Take one down, patch it around.

127 little bugs in the code…

From the old to the new

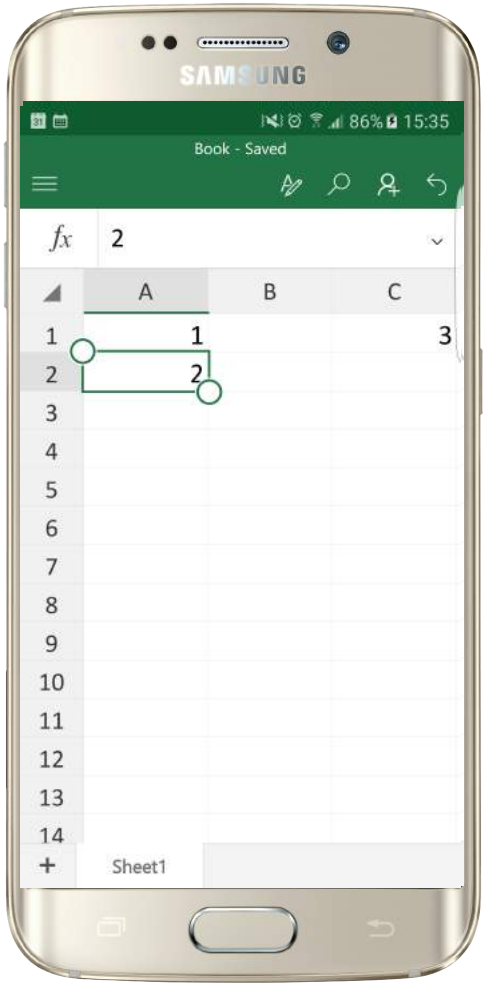


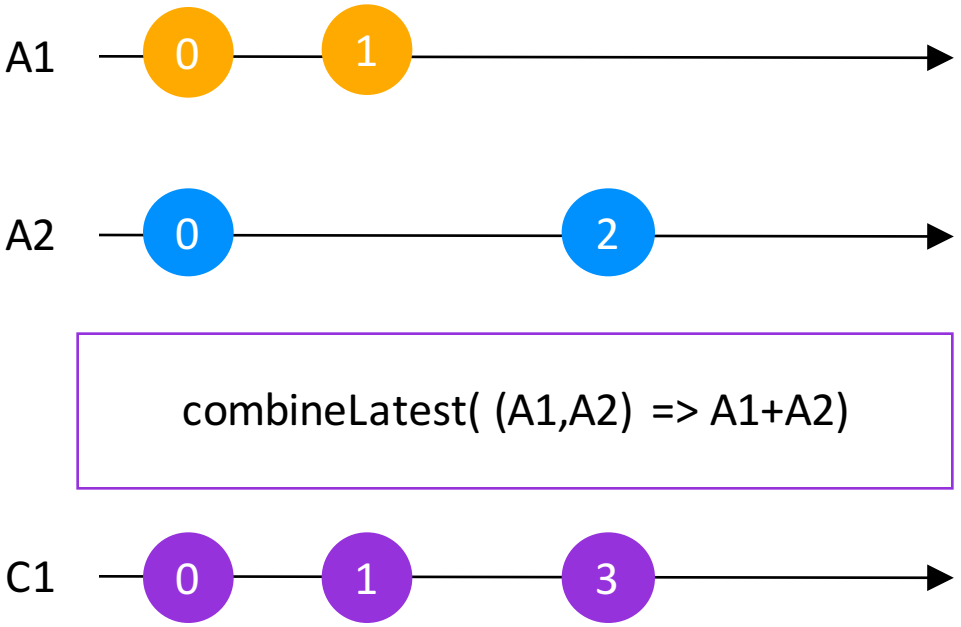
upday

Crash course in RxJava

upday

Sum it up!



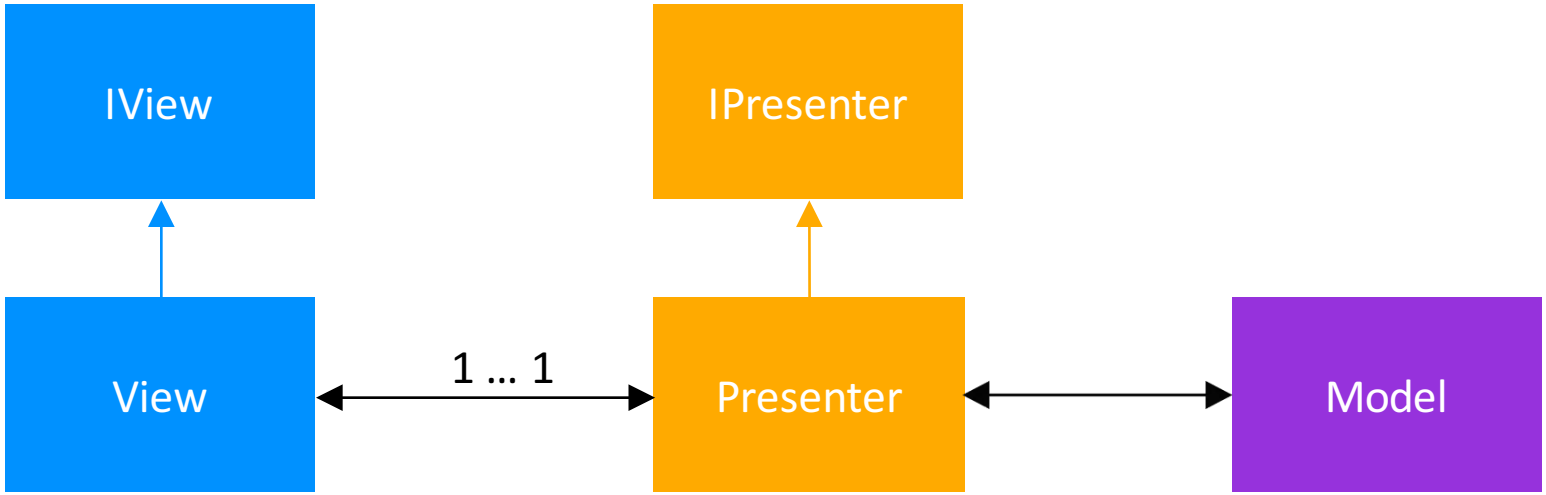


upday

What is MVVM and how do you apply it?

upday

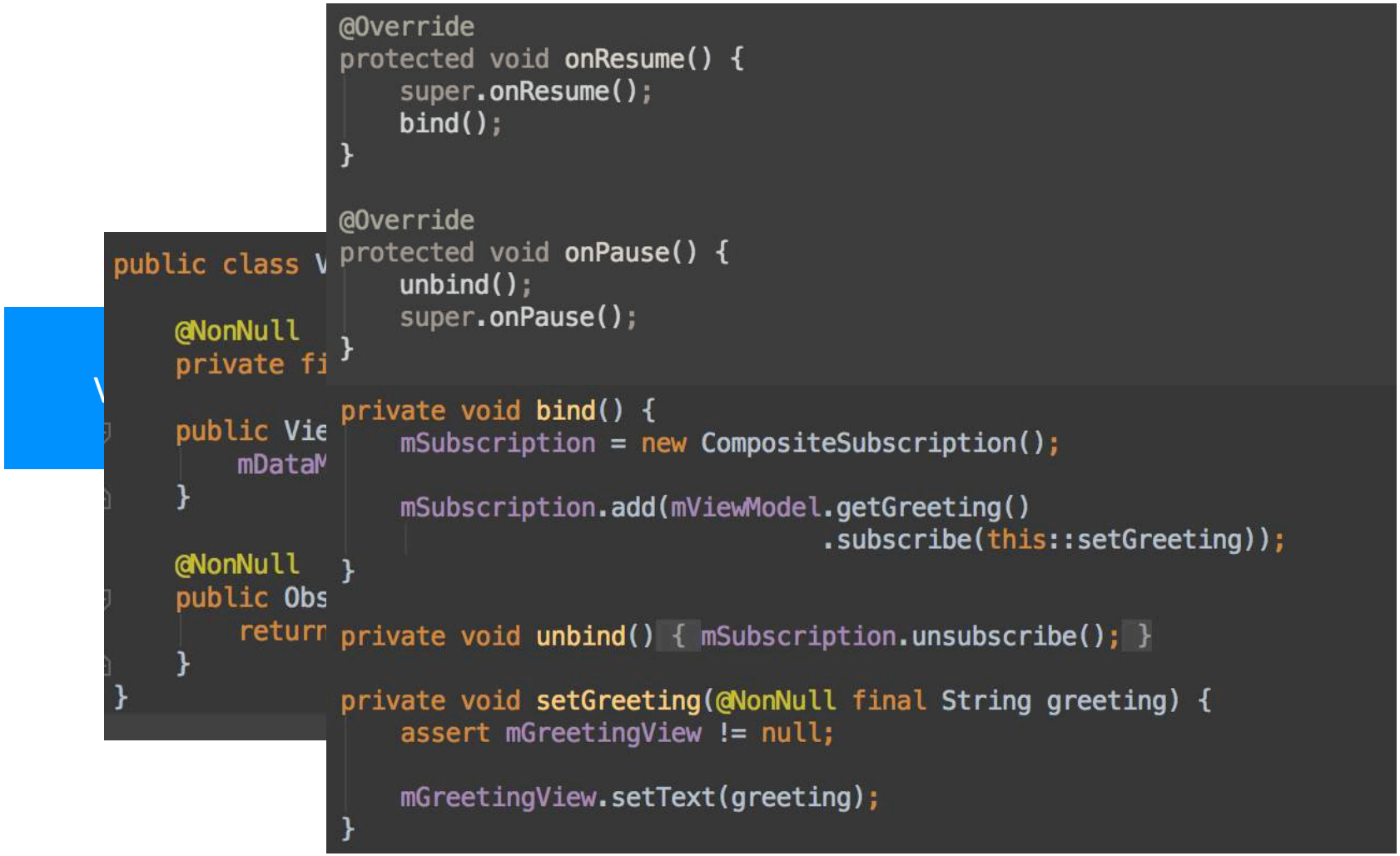
Model-View-Presenter



Model-View-ViewModel


SAMSUNG
76% 09:14
Droidcon Berlin
Hello, World!

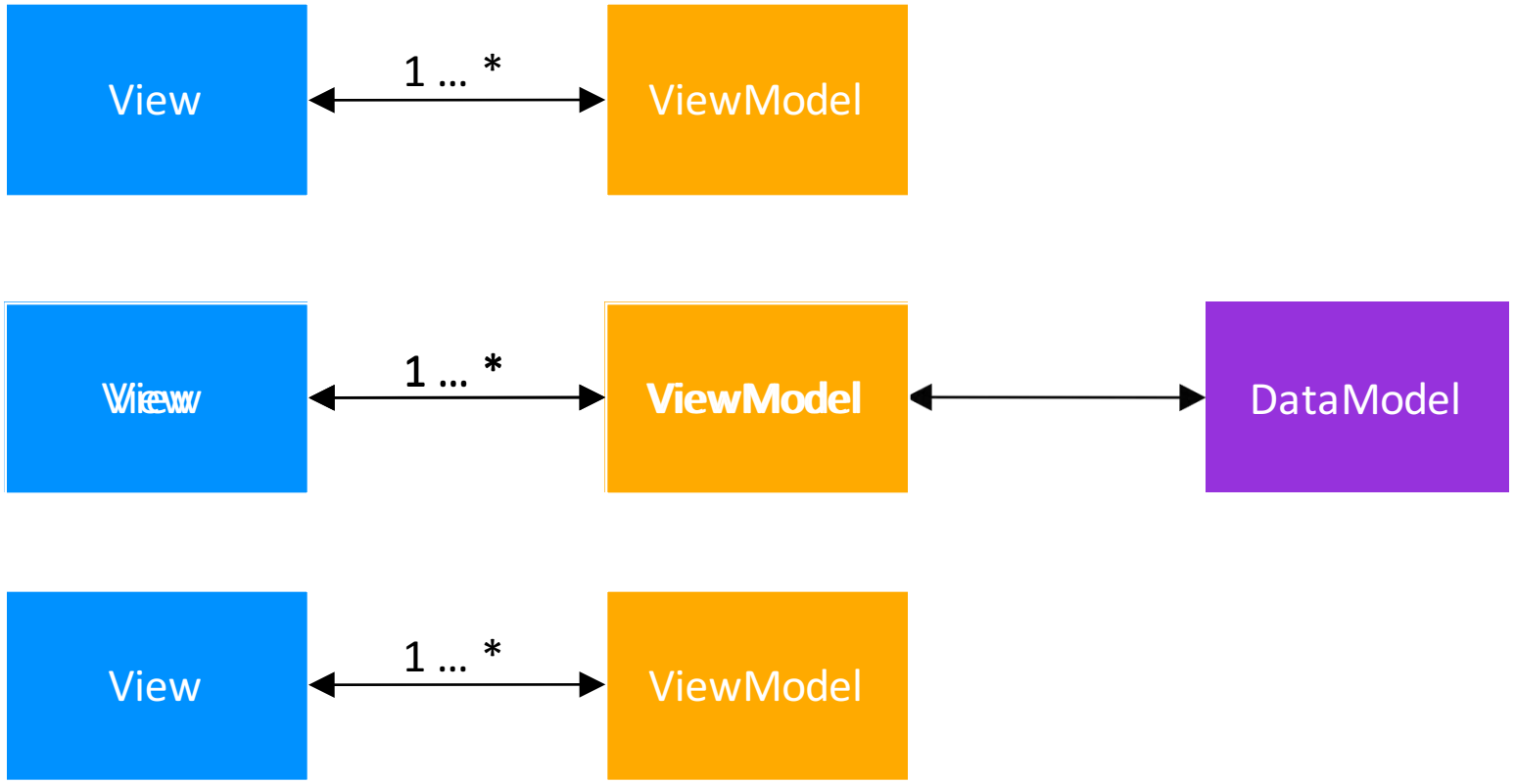
Model-View-ViewModel



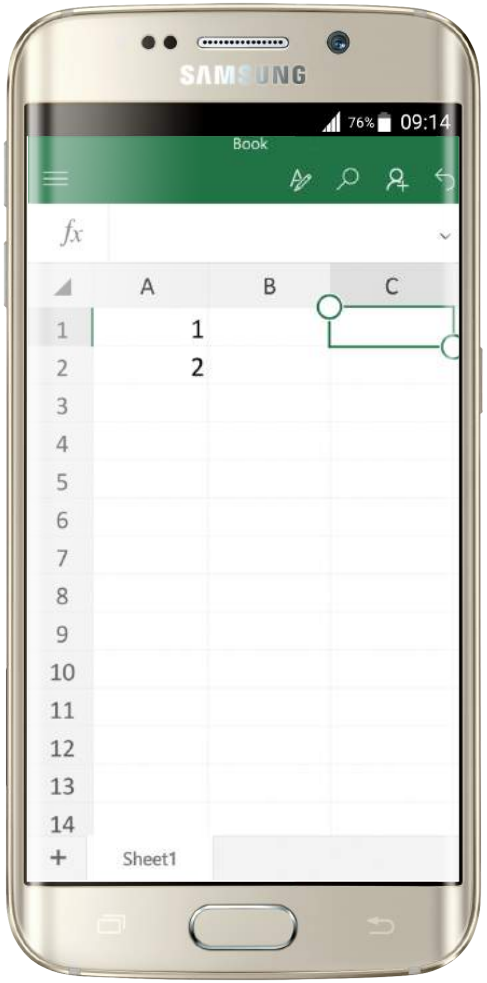
Testing

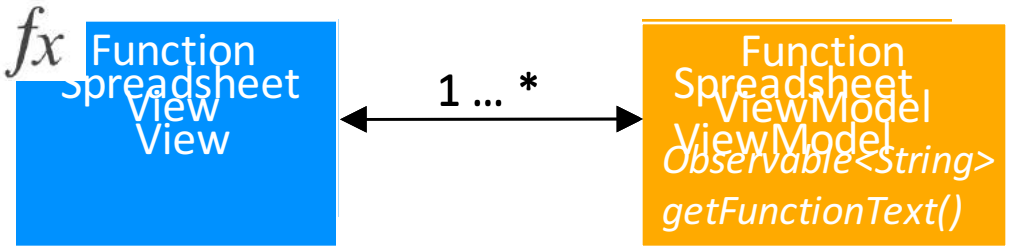
```
@Mock
private IDataModel mDataModel;

private ViewModel mViewModel;

@Before
public void setUp() throws Exception {
    MockitoAnnotations.initMocks(this);

    mViewModel = new ViewModel(mDataModel);
}

@Test
public void testGetGreeting_emitsCorrectGreeting() {
    String greeting = "Hello!";
    Mockito.when(mDataModel.getGreeting()).thenReturn(Observable.just(greeting));
    TestSubscriber<String> testSubscriber = new TestSubscriber<>();

    mViewModel.getGreeting().subscribe(testSubscriber);

    testSubscriber.assertValue(greeting);
}
```

Model-View-ViewModel



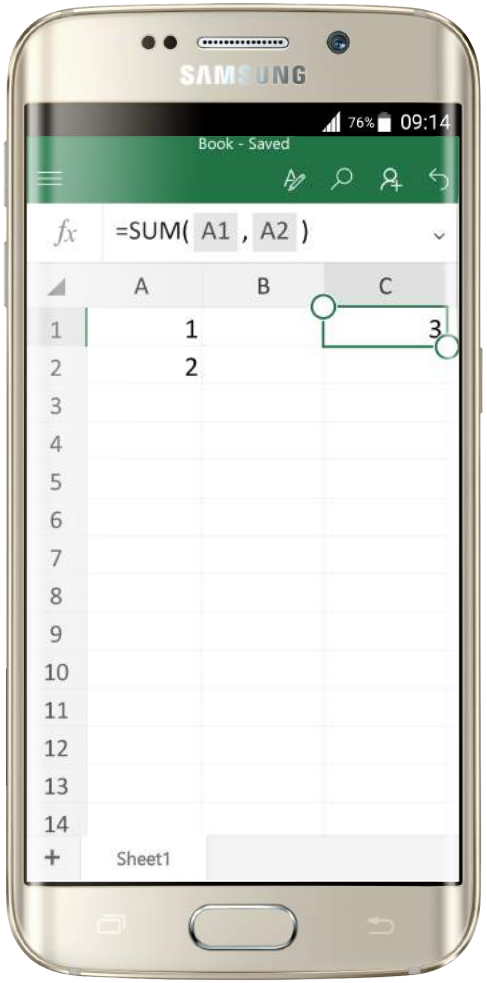
What is MVVM and how do you apply it?

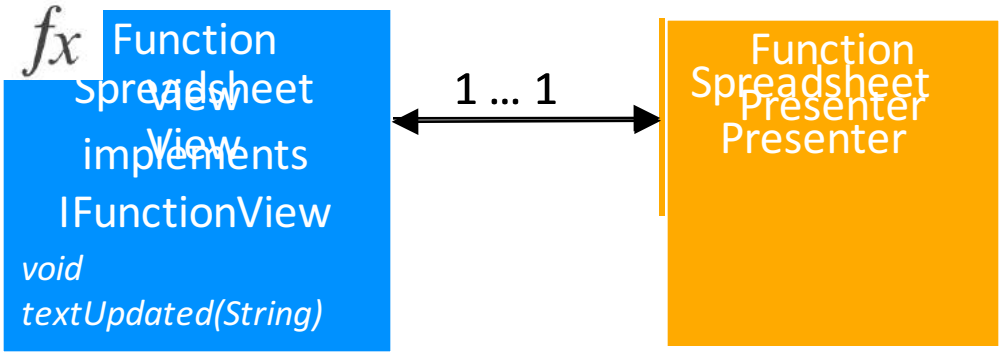


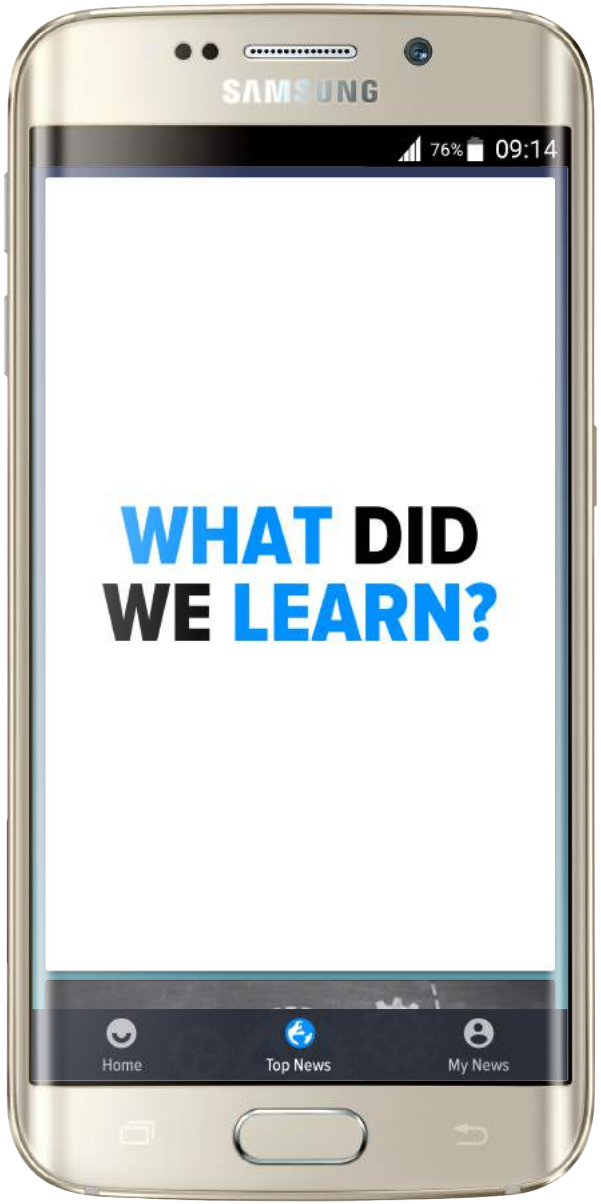


upday

What is MVVM and how do you apply it?





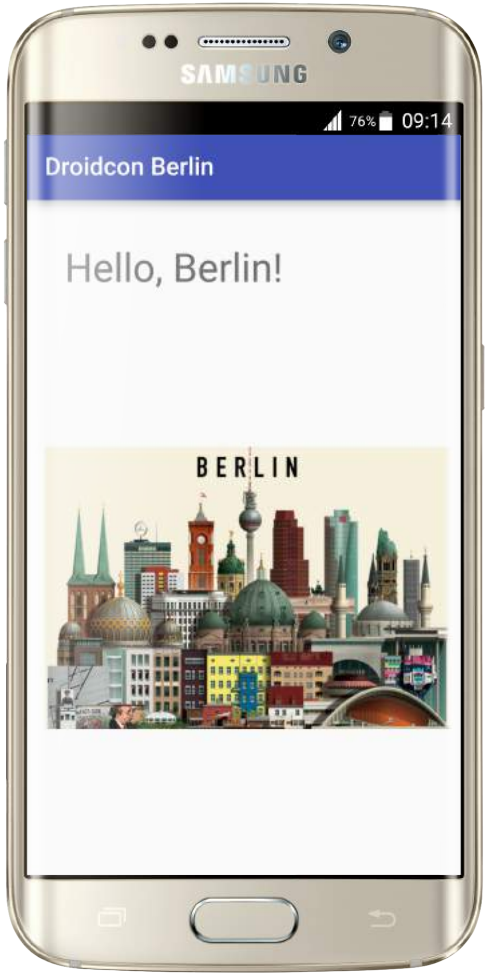


upday

We made mistakes but we recovered!

upday

We made mistakes but we recovered!



View

```
private void setGreeting(@NonNull final String greeting) {
    assert mGreetingView != null;

    mGreetingView.setText(greeting);
}
```

View

```
private void setGreeting(@NonNull final String greeting) {
    assert mGreetingView != null;

    mGreetingView.setText(greeting);

    if(greeting.contains("Berlin")){
        setImage(R.drawable.berlin);
    }
}

private void setImage(@DrawableRes final int resId) {
    assert mImageView != null;

    mImageView.setImageResource(resId);
}
```

ViewModel

```
@NonNull
public Observable<String> getGreeting() {
    return mDataModel.getGreeting();
}

@NonNull
public Observable<Integer> getImage() {
    return mDataModel.getGreeting()
                     .filter(greeting -> greeting.contains(BERLIN))
                     .map(__ -> R.drawable.berlin);
}
```

Testing the ViewModel – positive case

```
@NonNull
public Observable<Integer> getImage() {
    return mDataModel.getGreeting()
                     .filter(greeting -> greeting.contains(BERLIN))
                     .map(__ -> R.drawable.berlin);
}
```

```
@Test
public void testGetImage_emits_WhenGreetingContainsBerlin() {
    String greeting = "Hello " + ViewModel.BERLIN;
    Mockito.when(mDataModel.getGreeting()).thenReturn(Observable.just(greeting));
    TestSubscriber<Integer> testSubscriber = new TestSubscriber<>();

    mViewModel.getImage().subscribe(testSubscriber);

    testSubscriber.assertValue(R.drawable.berlin);
}
```

Testing the ViewModel – negative case

```
@NonNull
public Observable<Integer> getImage() {
    return mDataModel.getGreeting()
                     .filter(greeting -> greeting.contains(BERLIN))
                     .map(__ -> R.drawable.berlin);
}
```

```
@Test
public void testGetImage_doesNotEmit_WhenGreetingDoesNotContainBerlin() {
    String greeting = "Hello, World!";
    Mockito.when(mDataModel.getGreeting()).thenReturn(Observable.just(greeting));
    TestSubscriber<Integer> testSubscriber = new TestSubscriber<>();

    mViewModel.getImage().subscribe(testSubscriber);

    testSubscriber.assertNoValues();
}
```

View

```
private void bind() {
    mSubscription = new CompositeSubscription();

    mSubscription.add(mViewModel.getGreeting()
                                .subscribeOn(Schedulers.computation())
                                .observeOn(AndroidSchedulers.mainThread())
                                .subscribe(this::setGreeting));

    mSubscription.add(mViewModel.getImage()
                                .subscribeOn(Schedulers.computation())
                                .observeOn(AndroidSchedulers.mainThread())
                                .subscribe(this::setImage));
}

private void setGreeting(@NonNull final String greeting) {
    assert mGreetingView != null;

    mGreetingView.setText(greeting);
}

private void setImage(@DrawableRes final int resId) {
    assert mImageView != null;

    mImageView.setImageResource(resId);
}
```

What about Views that don’t have have lifecycle events?

```java
@Override
protected void onResume() {
    super.onResume();
    bind();
}

@Override
protected void onPause() {
    unbind();
    super.onPause();
}
```

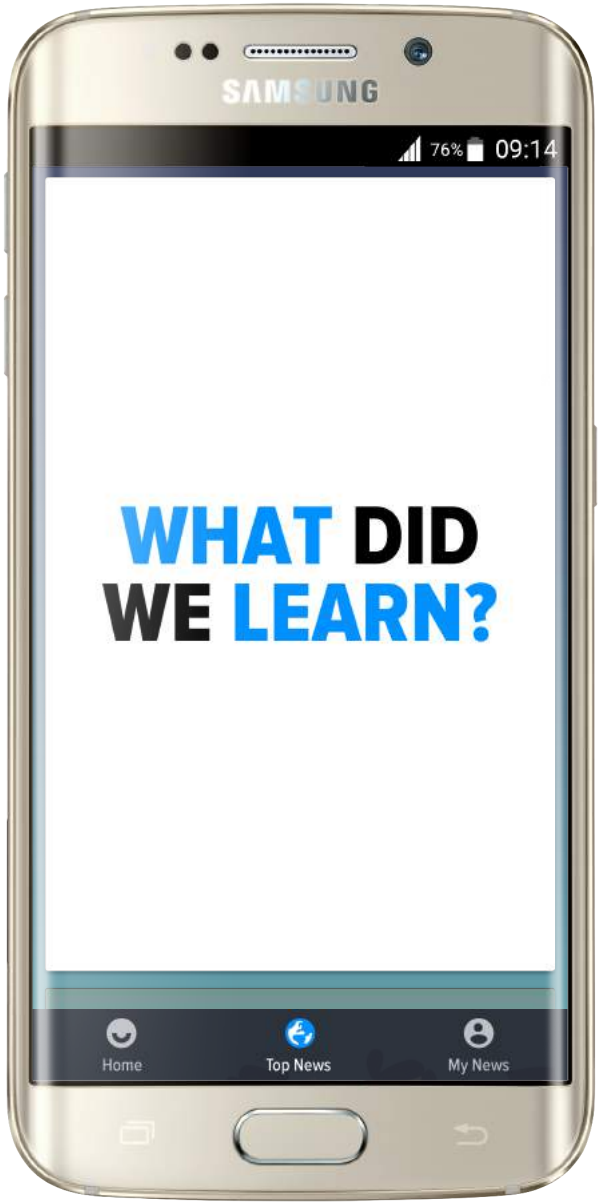
```java
private void bind() {
    mSubscription = new CompositeSubscription();

    mSubscription.add(mViewModel.getGreeting()
                                .subscribe(this::setGreeting));
}

private void unbind() { mSubscription.unsubscribe(); }

private void setGreeting(@NonNull final String greeting) {
    assert mGreetingView != null;

    mGreetingView.setText(greeting);
}
```

Views and subscriptions

```
public class MyView extends LinearLayout {

    @NonNull
    private CompositeSubscription mSubscription;

    @NonNull
    private ViewModel mViewModel;

    @Nullable
    private TextView mGreetingView;

    public MyView(@NonNull final Context context,
                  @NonNull final ViewModel viewModel) {
        super(context);
        mSubscription = new CompositeSubscription();
        mViewModel = viewModel;

        View.inflate(context, R.layout.view_layout, this);
        mGreetingView = (TextView) findViewById(R.id.greeting);

        bind();
    }

    private void bind() {
        mSubscription.add(mViewModel.getGreeting()
                                    .subscribe(this::setGreeting));
    }

    private void setGreeting(@NonNull final String greeting) {
        assert mGreetingView != null;

        mGreetingView.setText(greeting);
    }
```

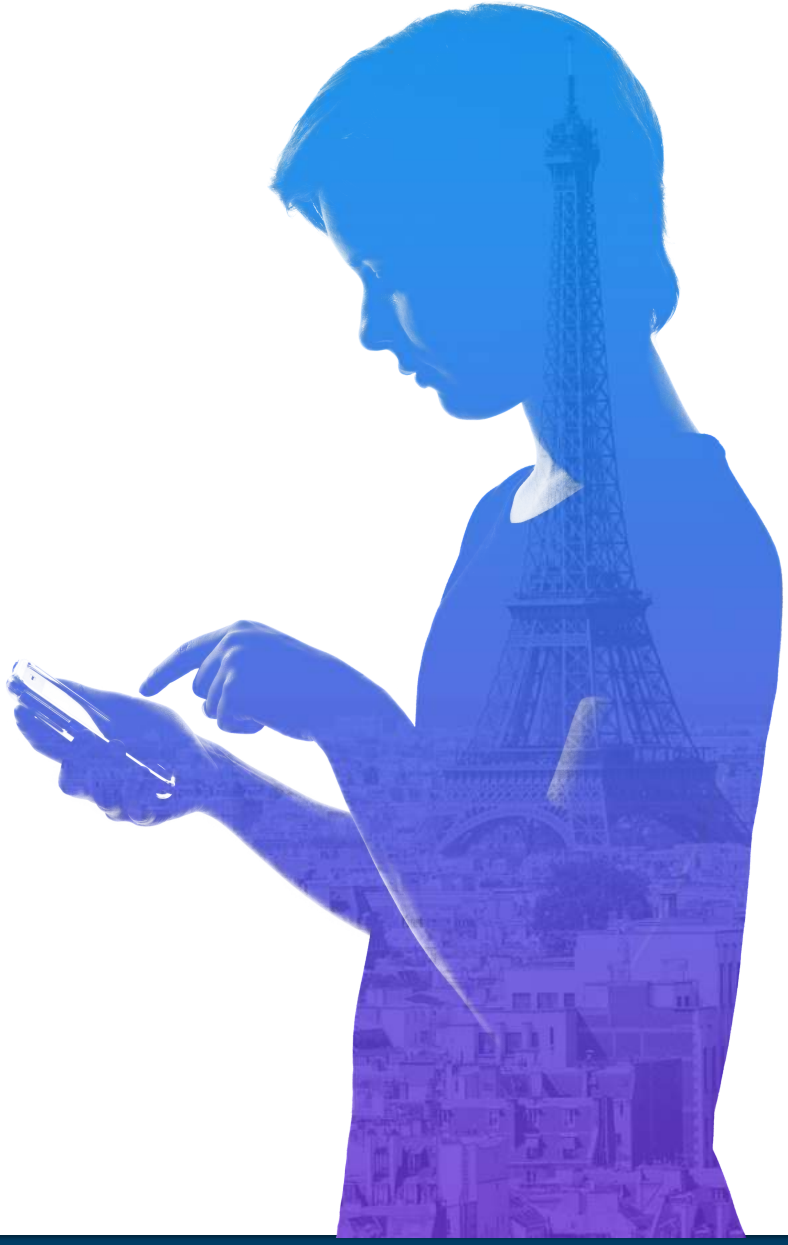
Views and subscriptions

```
private void unbind() { mSubscription.clear(); }

@Override
protected void onDetachedFromWindow() {
    unbind();
    super.onDetachedFromWindow();
}
```



upday

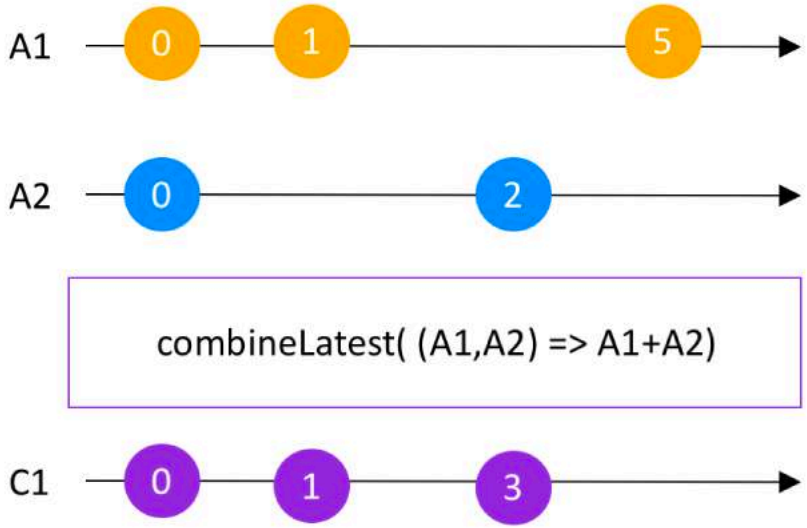
RxJava – the good and the bad



RxJava vs Threads/AsyncTask/Event bus frameworks

```
@NonNull
public Observable<Integer> getImage() {
    return mDataModel.getGreeting()
                     .filter(greeting -> greeting.contains(BERLIN))
                     .map(__ -> R.drawable.berlin);
}
```

Stream composability FTW!



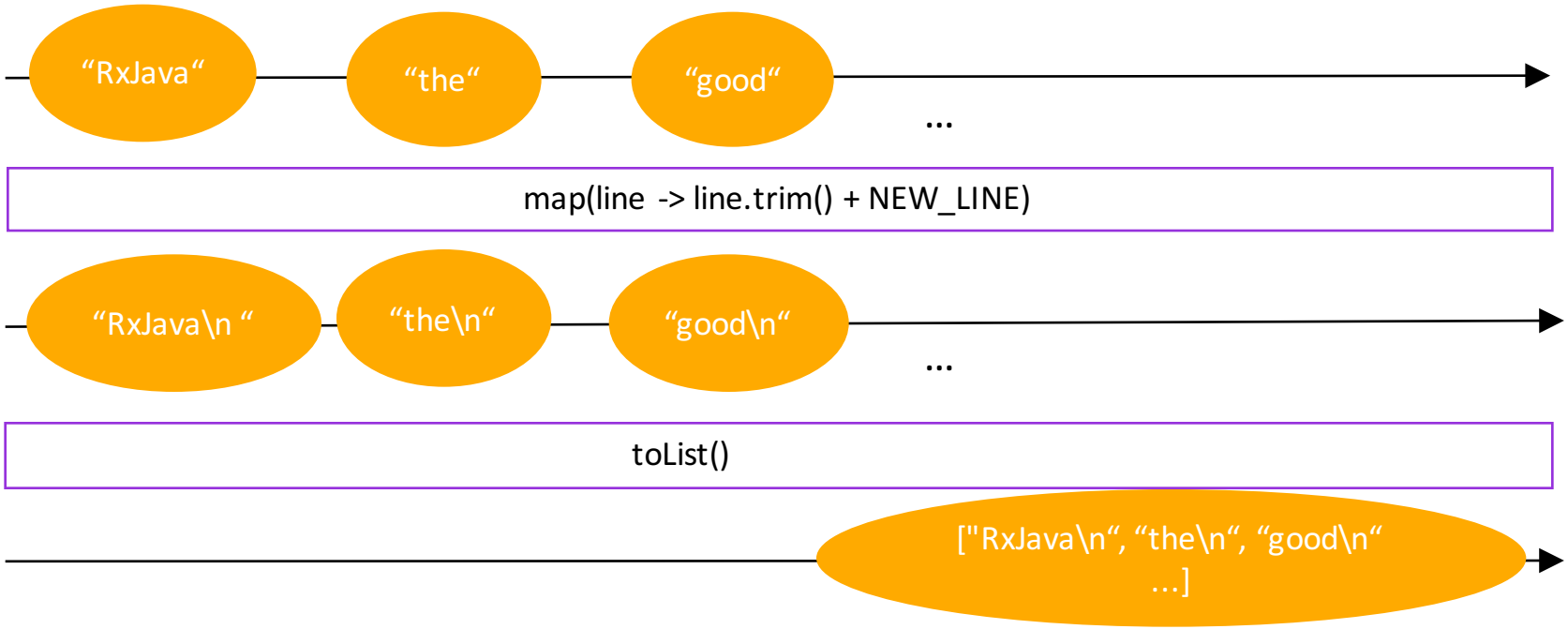

```
@NonNull
public Observable<Integer> getSum(@NonNull final Observable<Integer> a1,
                                  @NonNull final Observable<Integer> a2) {
    return Observable.combineLatest(a1,
                                    a2,
                                    (value1, value2) -> value1 + value2);
}
```

Split a string using a regular expression and append a new line to each item in the resulting list

“RxJava – the good and the bad”

[“RxJava\n”, “the\n”, “good\n”, ”and\n”, “the\n”, ”bad\n”]

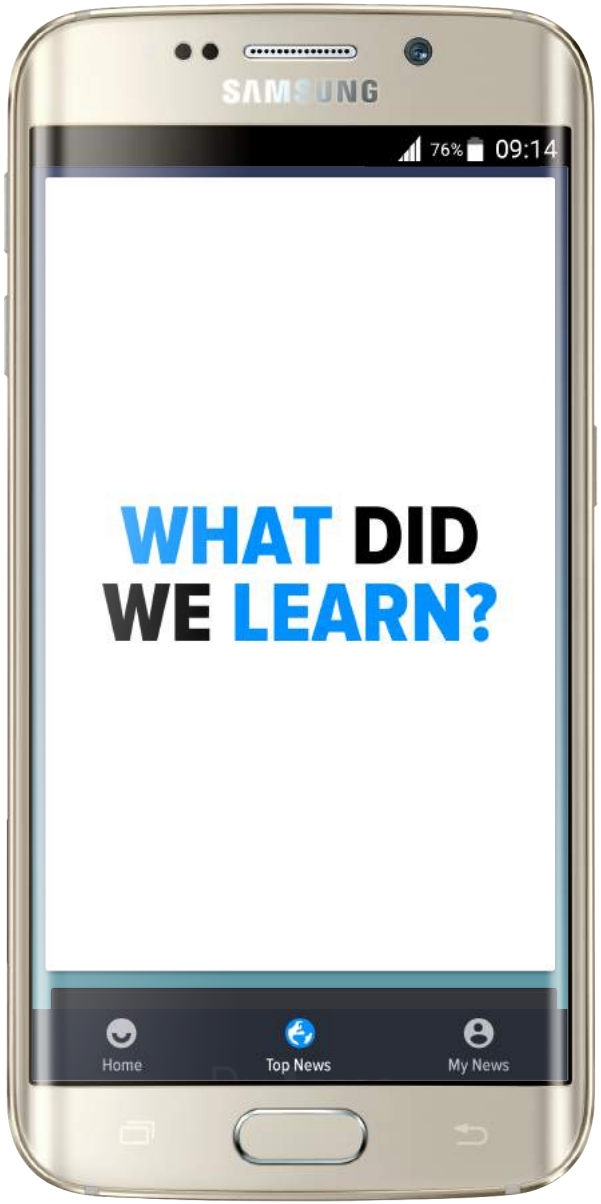
```
private static List<String> splitDeclarative(@NonNull final String text) {
    return Observable.from(text.split(SEPARATOR))
                     .map(line -> line.trim() + NEW_LINE)
                     .toList()
                     .toBlocking()
                     .single();
}
```



Split a string using a regular expression and append a new line to each item in the resulting list

```
private static List<String> splitDeclarative(@NonNull final String text) {
    return Observable.from(text.split(SEPARATOR))
                     .map(line -> line.trim() + NEW_LINE)
                     .toList()
                     .toBlocking()
                     .single();
}
```

```
private static List<String> splitIterative(@NonNull final String text) {
    String[] splitString = text.split(SEPARATOR);
    List<String> strings = new ArrayList<>();
    for (final String line : splitString) {
        strings.add(line.trim() + NEW_LINE);
    }
    return strings;
}
```



upday

RxJava and MVVM the perfect mix

Exclusive to Samsung

Simple MVVM example: https://github.com/florina-muntenescu/DroidconMVVM
MVP vs MVVM example: https://github.com/florina-muntenescu/MVPvsMVVM

upday tech blog: **https://upday.github.io/**

upday