ANDROID ANIMATION:

FROM HATE TO LOVE

- VIEW ANIMATION
- PROPERTY ANIMATION
- TRANSITIONS
- MOTION LAYOUT

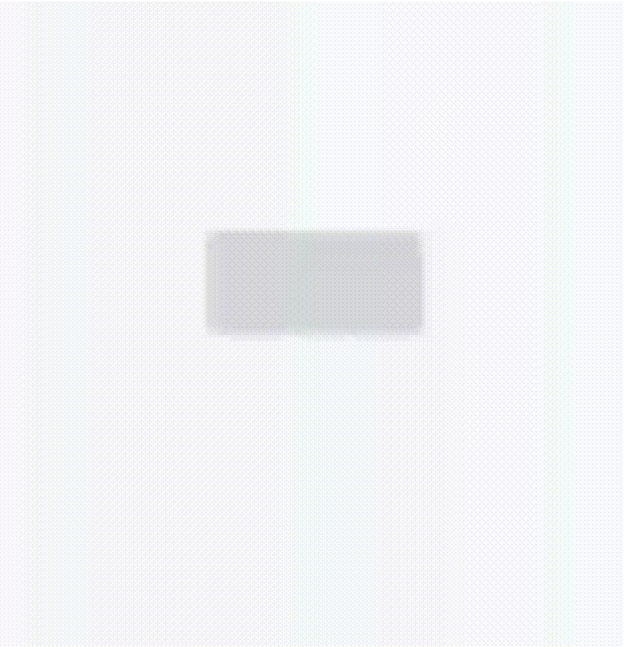
Алексей Зотов

GITHUB со всеми анимациями

VIEW ANIMATION

```
<translate xmlns:android="http://schemas.android.com/apk/res/android"
    android:duration="250"
    android:fillAfter="true"
    android:fromYDelta="0"
    android:toYDelta="200" />
```

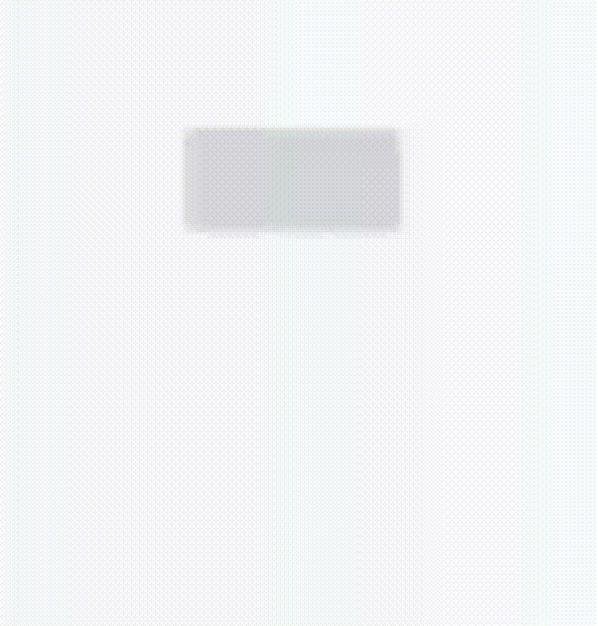
```
private fun startAnimation(view: View) {
    val animation = AnimationUtils.loadAnimation(this, R.anim.view_animation)
    view.startAnimation(animation)
}
```



PROPERTY ANIMATOR

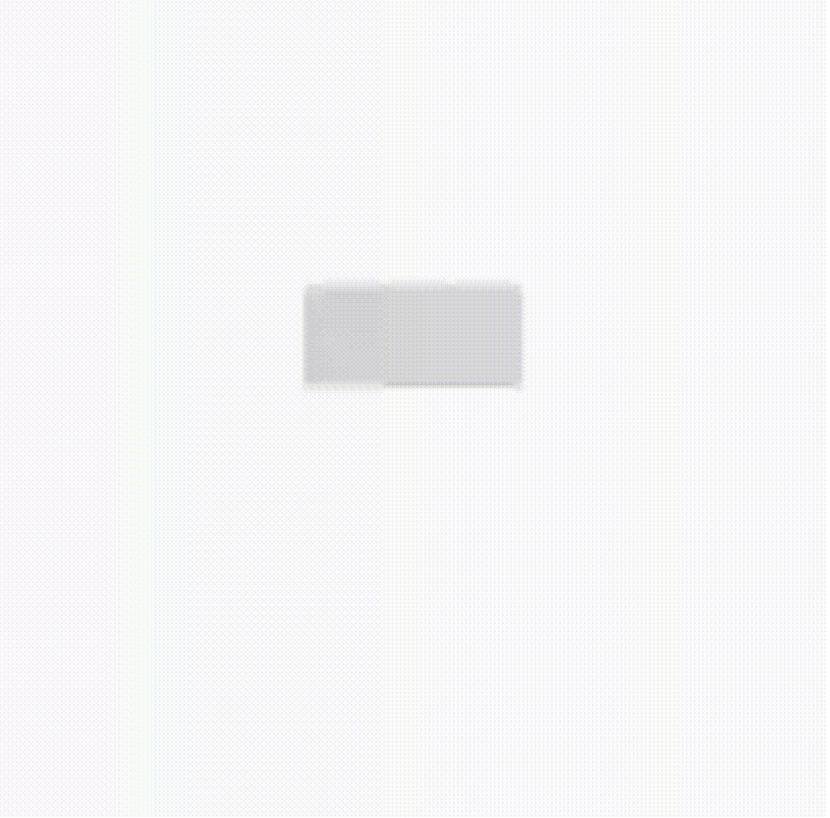
```
private fun startAnimation(view: View) {
    view.animate()
        .translationY(200f)
        .setDuration(250)
        .start()
}
```

```
<selector xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:state_enabled="true">
        <set android:ordering="together">
            <objectAnimator
                android:duration="250"
                android:interpolator="@android:anim/decelerate_interpolator"
                android:propertyName="translateY"
                android:valueTo="200"
                android:valueType="floatType" />
        </set>
    </item>
    <item android:state_enabled="false">
        <set android:ordering="together">
            <objectAnimator
                android:duration="250"
                android:interpolator="@android:anim/accelerate_interpolator"
                android:propertyName="translateY"
                android:valueTo="0"
                android:valueType="floatType" />
        </set>
    </item>
</selector>
```



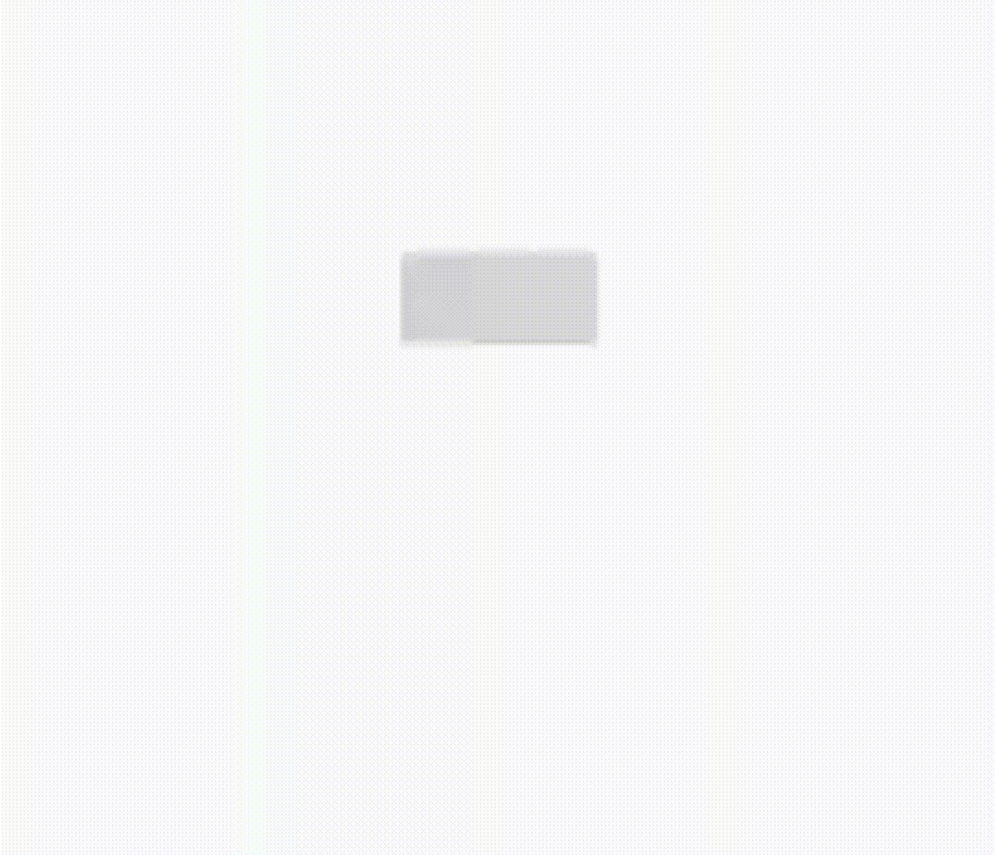
PROPERTY ANIMATOR

```
private fun startAnimation(view: View) {
    view.animate()
        .translationY(200f)
        .rotation(180f)
        .setDuration(400)
        .start()
}
```



PROPERTY ANIMATOR

```
private fun startAnimation(view: View) {
    view.animate()
        .translationY(200f)
        .rotation(180f)
        .scaleY(3f)
        .scaleX(4f)
        .setDuration(400)
        .start()
}
```



TRANSITION

```
Scene.getSceneForLayout(main_container, R.layout.scene_a, this)
```

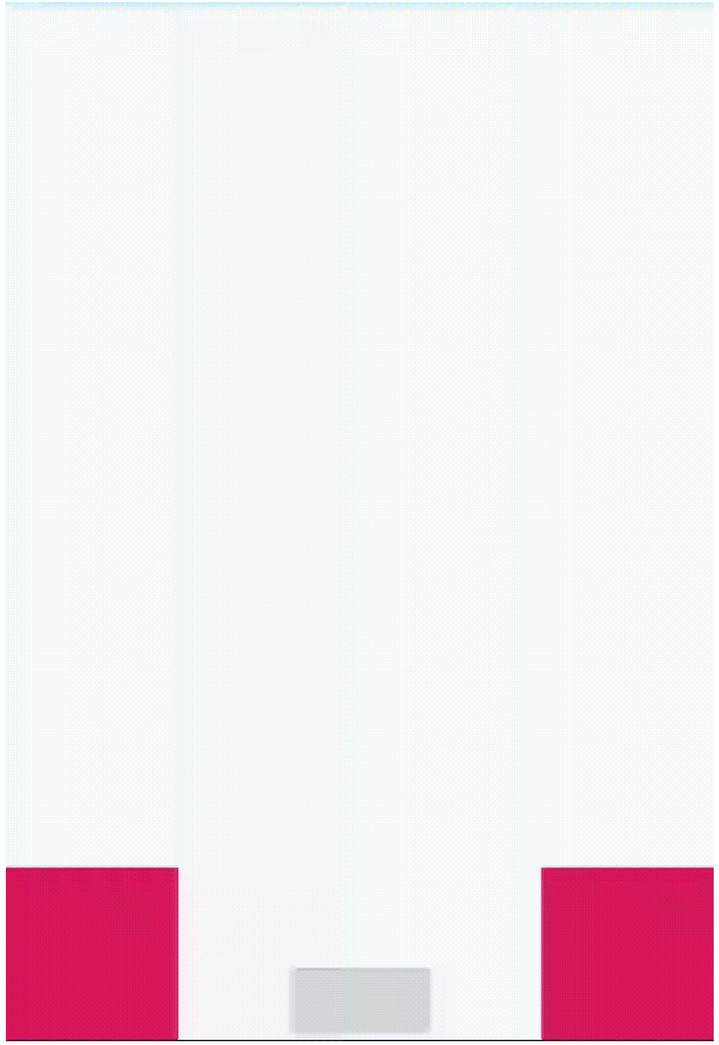
```
TransitionManager.beginDelayedTransition(main_container)
```

```
TransitionManager.go(scene)
```

```
private val startSet = ConstraintSet()
private val endSet = ConstraintSet()
```

```
startSet.clone(main_container)
endSet.clone(this, R.layout.scene_a)
```

```
private fun startAnimation(reverse: Boolean) {
    TransitionManager.beginDelayedTransition(main_container)
    (if (reverse) startSet else endSet).applyTo(main_container)
}
```



TRANSITION

```
android:transitionName="@string/cat_transition"
```

```
private fun openCatActivity(cat: View) {
    val bundle = ActivityOptions.makeSceneTransitionAnimation(
        this,
        cat,
        getString(R.string.cat_transition)
    ).toBundle()

    startActivity(
        Intent(this, CatInfoActivity::class.java),
        bundle
    )
}
```

TRANSITION

```
<transitionSet xmlns:android="http://schemas.android.com/apk/res/android"
    android:transitionOrdering="together">
    <changeBounds />
    <changeClipBounds />
    <changeTransform />
    <changeImageTransform />
</transitionSet>
```

```
<style name="CatActivity" parent="Theme.AppCompat.Light.NoActionBar">
    <item name="android:windowActivityTransitions">true</item>

    <item name="android:windowSharedElementEnterTransition">
        @transition/cat_open
    </item>

    <item name="android:windowSharedElementExitTransition">
        @transition/cat_open
    </item>
</style>
```

```
<activity
    android:name=".CatInfoActivity"
    android:theme="@style/CatActivity" />
```

MOTION LAYOUT

```
<androidx.constraintlayout.motion.widget.MotionLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/motion_container"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    app:layoutDescription="@xml/scene"
    tools:showPaths="true">

    <View
        android:id="@+id/view"
        android:layout_width="64dp"
        android:layout_height="64dp"
        android:background="@color/colorAccent"
        tools:text="Button" />

</androidx.constraintlayout.motion.widget.MotionLayout>
```

```
<Transition
    motion:constraintSetEnd="@+id/end"
    motion:constraintSetStart="@+id/start"
    motion:duration="1000">
    <OnSwipe
        motion:dragDirection="dragRight"
        motion:touchAnchorId="@+id/view"
        motion:touchAnchorSide="right" />
</Transition>
```

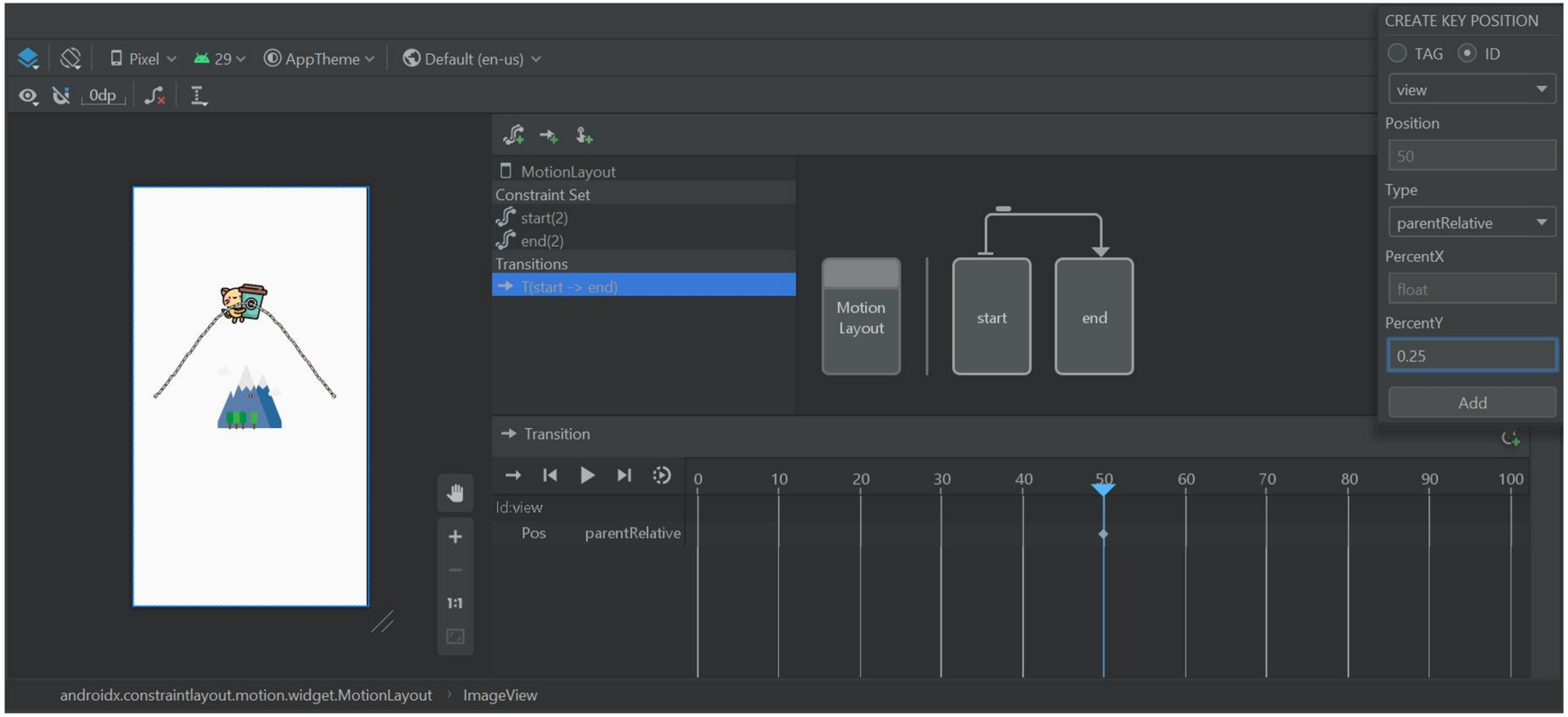
```
<ConstraintSet android:id="@+id/start">
    <Constraint
        android:id="@+id/view"
        android:layout_width="64dp"
        android:layout_height="64dp"
        android:layout_marginStart="8dp"
        motion:layout_constraintBottom_toBottomOf="parent"
        motion:layout_constraintStart_toStartOf="parent"
        motion:layout_constraintTop_toTopOf="parent" />
</ConstraintSet>

<ConstraintSet android:id="@+id/end">
    <Constraint
        android:id="@+id/view"
        android:layout_width="64dp"
        android:layout_height="64dp"
        android:layout_marginEnd="8dp"
        motion:layout_constraintBottom_toBottomOf="parent"
        motion:layout_constraintEnd_toEndOf="parent"
        motion:layout_constraintTop_toTopOf="parent" />
</ConstraintSet>
```

MOTION LAYOUT

```
<MotionScene xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:motion="http://schemas.android.com/apk/res-auto">

    <Transition
        motion:constraintSetEnd="@+id/end"
        motion:constraintSetStart="@+id/start"
        motion:duration="1000">
        <OnSwipe
            motion:dragDirection="dragRight"
            motion:touchAnchorId="@+id/view"
            motion:touchAnchorSide="right" />
    </Transition>

    <ConstraintSet android:id="@+id/start">
        <Constraint
            android:id="@+id/view"
            android:layout_width="64dp"
            android:layout_height="64dp"
            android:layout_marginStart="8dp"
            motion:layout_constraintBottom_toBottomOf="parent"
            motion:layout_constraintStart_toStartOf="parent"
            motion:layout_constraintTop_toTopOf="parent" />
    </ConstraintSet>

    <ConstraintSet android:id="@+id/end">
        <Constraint
            android:id="@+id/view"
            android:layout_width="64dp"
            android:layout_height="64dp"
            android:layout_marginEnd="8dp"
            motion:layout_constraintBottom_toBottomOf="parent"
            motion:layout_constraintEnd_toEndOf="parent"
            motion:layout_constraintTop_toTopOf="parent" />
    </ConstraintSet>

</MotionScene>
```

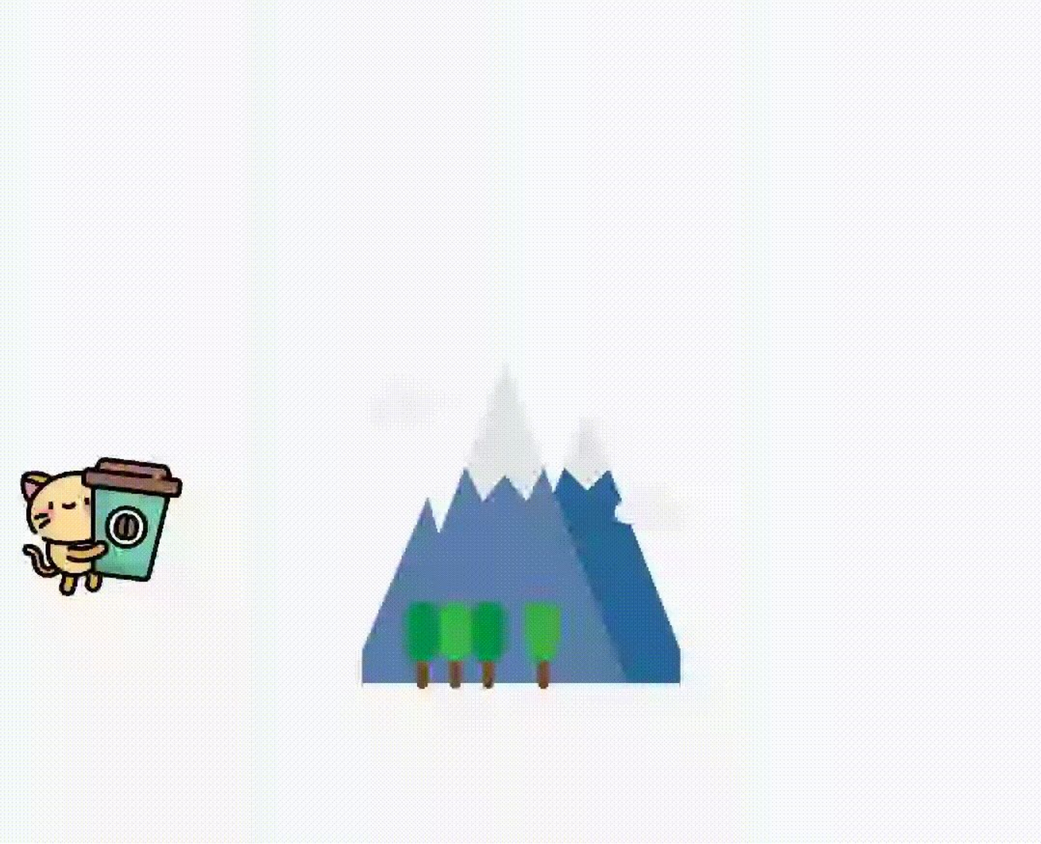
MOTION LAYOUT

```
<KeyPosition
    motion:framePosition="50"
    motion:keyPositionType="parentRelative"
    motion:motionTarget="@+id/view"
    motion:percentY="0.25" />
```



MOTION LAYOUT

```
<KeyPosition
    motion:framePosition="50"
    motion:keyPositionType="parentRelative"
    motion:motionTarget="@+id/view"
    motion:percentY="0.25" />
```



MOTION LAYOUT

```
<androidx.constraintlayout.motion.widget.MotionLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/motion_container"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    app:layoutDescription="@xml/scene"
    tools:showPaths="true">

    <View
        android:id="@+id/view"
        android:layout_width="64dp"
        android:layout_height="64dp"
        android:background="@color/colorAccent"
        tools:text="Button" />

</androidx.constraintlayout.motion.widget.MotionLayout>
```

```
<Transition
    motion:constraintSetEnd="@+id/end"
    motion:constraintSetStart="@+id/start"
    motion:duration="1000">
    <OnSwipe
        motion:dragDirection="dragRight"
        motion:touchAnchorId="@+id/view"
        motion:touchAnchorSide="right" />
</Transition>
```

```
<ConstraintSet android:id="@+id/start">
    <Constraint
        android:id="@+id/view"
        android:layout_width="64dp"
        android:layout_height="64dp"
        android:layout_marginStart="8dp"
        motion:layout_constraintBottom_toBottomOf="parent"
        motion:layout_constraintStart_toStartOf="parent"
        motion:layout_constraintTop_toTopOf="parent" />
</ConstraintSet>

<ConstraintSet android:id="@+id/end">
    <Constraint
        android:id="@+id/view"
        android:layout_width="64dp"
        android:layout_height="64dp"
        android:layout_marginEnd="8dp"
        motion:layout_constraintBottom_toBottomOf="parent"
        motion:layout_constraintEnd_toEndOf="parent"
        motion:layout_constraintTop_toTopOf="parent" />
</ConstraintSet>
```

VIEW ANIMATION	PROPERTY ANIMATOR
• Просто не используйте это. Никогда. Я серьезно	• самый простой в изучении • простая анимация 1 объекта • гибкий • анимация созависимых VIEW • Анимация большого количества VIEW
TRANSITIONS	**MOTION**
• анимация созависимых VIEW • анимация большого количества VIEW • кривые траектории • события во время анимации • OVERKILL для одной независимой VIEW	• все плюсы TRANSTITIONS • кривые траектории • зависимость анимаций от TIMELINE • MOTION EDITOR • OVERKILL для одной независимой VIEW • сложнее в освоении

GITHUB со всеми анимациями

Спасибо

Алексей зотов

Яндекс

Яндекс

itcommunity

REACTIVE APPROACH:

KEEP IT SIMPLE IN ANDROID

Даниэл Сергеев, AUTORU AN

Подходы к написанию приложений

- Императивный подход – парадигма программирования, ориентированная на последовательное выполнение команд, и внешних синхронных операций.
- **Реактивный подход** – парадигма программирования, ориентированная на потоки данных и асинхронное распространение изменений.

Реактивный подход

Эффективен

- В асинхронных приложениях
- Для обработки ошибок
- Для разгрузки MAIN THREAD

Недостатки

- Высокий порог вхождения
- Высокая сложность

Зачем реактивный подход? CALLBACK HELL

```
function hell(win) {
 // for listener purpose
 return function() {
   loadLink(win, REMOTE_SRC+'/assets/css/style.css', function() {
     loadLink(win, REMOTE_SRC+'/lib/async.js', function() {
       loadLink(win, REMOTE_SRC+'/lib/easyXDM.js', function() {
         loadLink(win, REMOTE_SRC+'/lib/json2.js', function() {
           loadLink(win, REMOTE_SRC+'/lib/underscode.min.js', function() {
             loadLink(win, REMOTE_SRC+'/lib/backbone.min.js', function() {
               loadLink(win, REMOTE_SRC+'/dev/base_dev.js', function() {
                 loadLink(win, REMOTE_SRC+'/assets/js/deps.js', function() {
                   loadLink(win, REMOTE_SRC+'/src/' + win.loader_path + '/loader.js', function() {
                     async.eachSeries(SCRIPTS, function(src, callback) {
                       loadScript(win, BASE_URL+src, callback);
                     });
                   });
                 });
               });
             });
           });
         });
       });
     });
   });
 };
}
```

Императивный подход

```
interface IUserManager {
    fun getUser(): User
    fun getUserBalance(userId: String): BigDecimal
    fun updateUserBalance(userId: String, balance: BigDecimal)
}

private val manager: IUserManager = UserManager()

override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_main)

    val user = manager.getUser()
    val balance = manager.getUserBalance(user.id)
    val newBalance = balance - payment
    manager.updateUserBalance(user.id, newBalance)

    view.showSnack("Balance has been updated to $newBalance")
```

```
interface IUserManager {
    fun getUser(onSuccess: (User) -> Unit, onError: (Throwable) -> Unit)
    fun getUserBalance(userId: String, onSuccess: (BigDecimal) -> Unit, onError: (Throwable) -> Unit)
    fun updateUserBalance(userId: String, balance: BigDecimal, onSuccess: () -> Unit, onError: (Throwable) -> Unit)
}

private val manager: IUserManager = UserManager()

override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_main)

    manager.getUser(
        onSuccess = { user ->
            manager.getUserBalance(
                userId = user.id,
                onSuccess = { balance ->
                    val newBalance = balance - payment
                    manager.updateUserBalance(
                        userId = user.id,
                        balance = newBalance,
                        onSuccess = { view.showSnack("balance updated") },
                        onError = ::processError
                    )
                },
                onError = ::processError
            )
        },
        onError = ::processError
    )
}

private fun processError(th: Throwable) { ... }
```

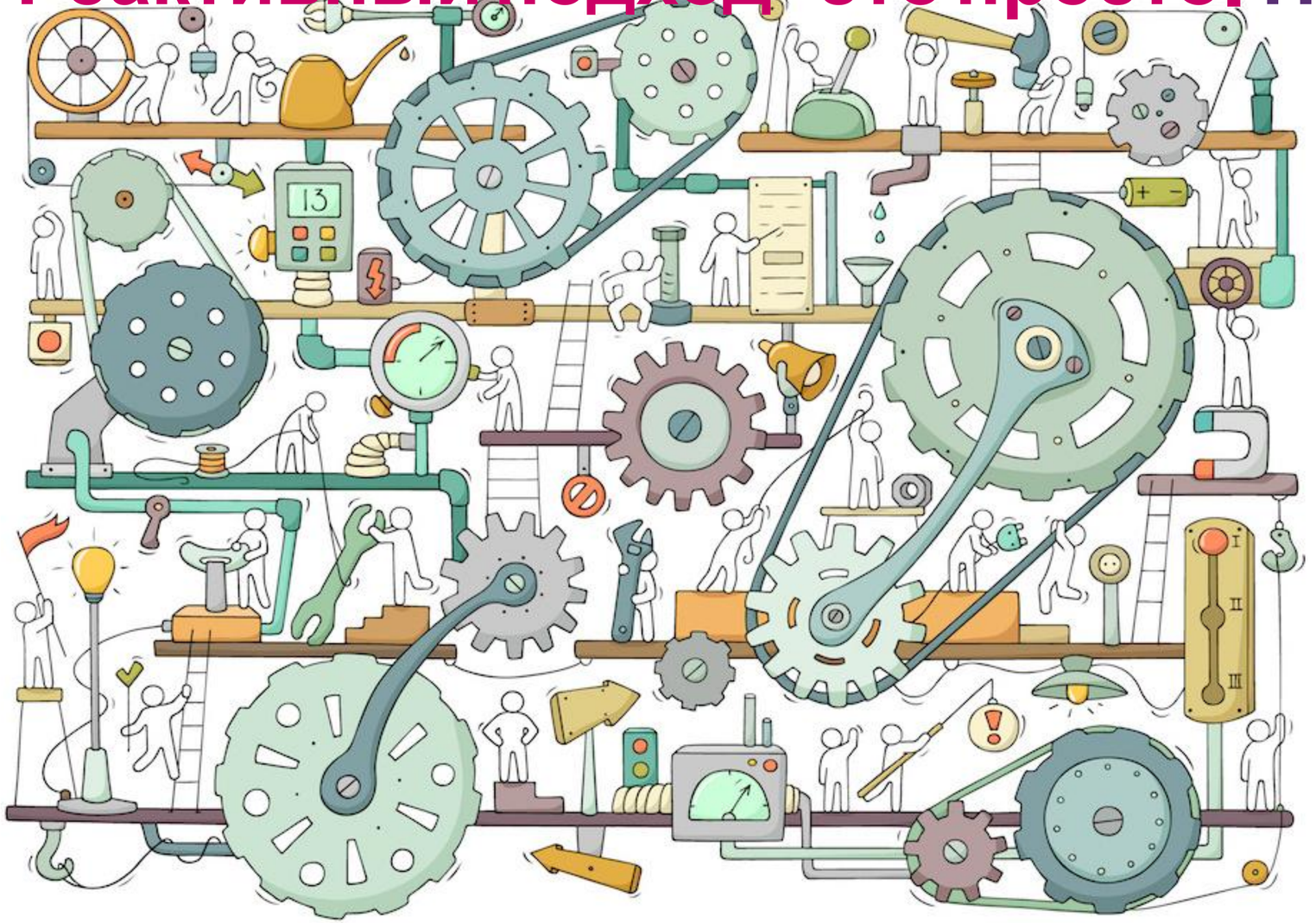
```
interface IUserManager {
    fun getUser(onSuccess: (User) -> Unit, onError: (Throwable) -> Unit)
    fun getUserBalance(userId: String, onSuccess: (BigDecimal) -> Unit, onError: (Throwable) -> Unit)
    fun updateUserBalance(userId: String, balance: BigDecimal, onSuccess: () -> Unit, onError: (Throwable) -> Unit)
}

private val manager: IUserManager = UserManager()

override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceS
    setContentView(R.layout

    manager.getUser(
        onSuccess =
            manager.  serB  ce(
                userI  ser.id,
                onSu  s = { balanc
                    va  Balance = bal  - payment
                    ma  er.updateUserBa  (
                        d = user.id,
                        ce = newBalance,
                        o  ess = { view.showSna  alan  dated") },
                        on  = ::processError
                    )
                },
                onError = ::proce
            )
        },
        onError = ::processError
    )
}

private fun processError(th: Throwable) { ... }
```

Слишком сложно

Реактивный подход

```
interface IUserManager {
    fun getUser(): Single<User>
    fun getUserBalance(userId: String): Single<BigDecimal>
    fun updateUserBalance(userId: String, balance: BigDecimal): Completable
}

private val manager: IUserManager = UserManager()

override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_main)

    manager.getUser()
        .flatMapCompletable { user ->
            manager.getUserBalance(user.id)
                .map { balance -> balance - cost }
                .flatMap { balance -> updateUserBalance(user.id, balance) }
        }
        .subscribe(            onSuccess = { view.showSnack("balance updated") },
            onError = ::processError
        )
}
```

Реактивный подход - это просто? Ну такое…



Реализация реактивного подхода в ANDROID

1. RXJAVA

HTTPS://GITHUB.COM/REACTIVEX/RXJAVA

2. COROUTINES

HTTPS://KOTLINLANG.ORG/DOCS/REFERENCE/COROUTINES-OVERVIEW.HTML

3. ANDROID LIVEDATA

HTTPS://DEVELOPER.ANDROID.COM/TOPIC/LIBRARIES/ARCHITECTURE/LIVEDATA

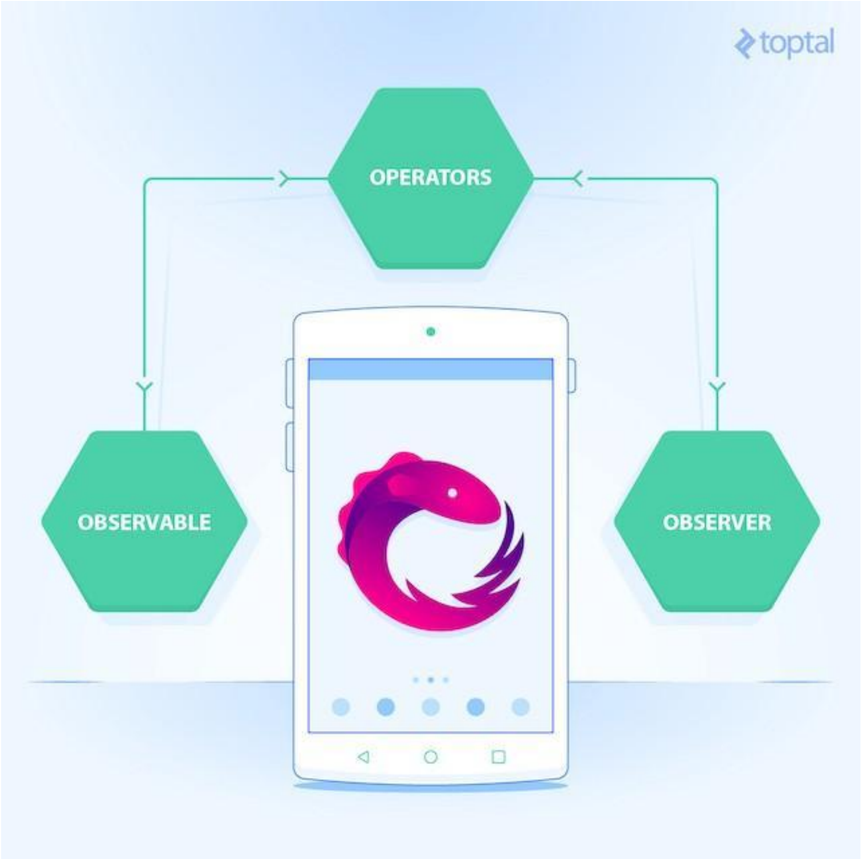
4. REACTOR, AKKA, ETS..

RXJAVA

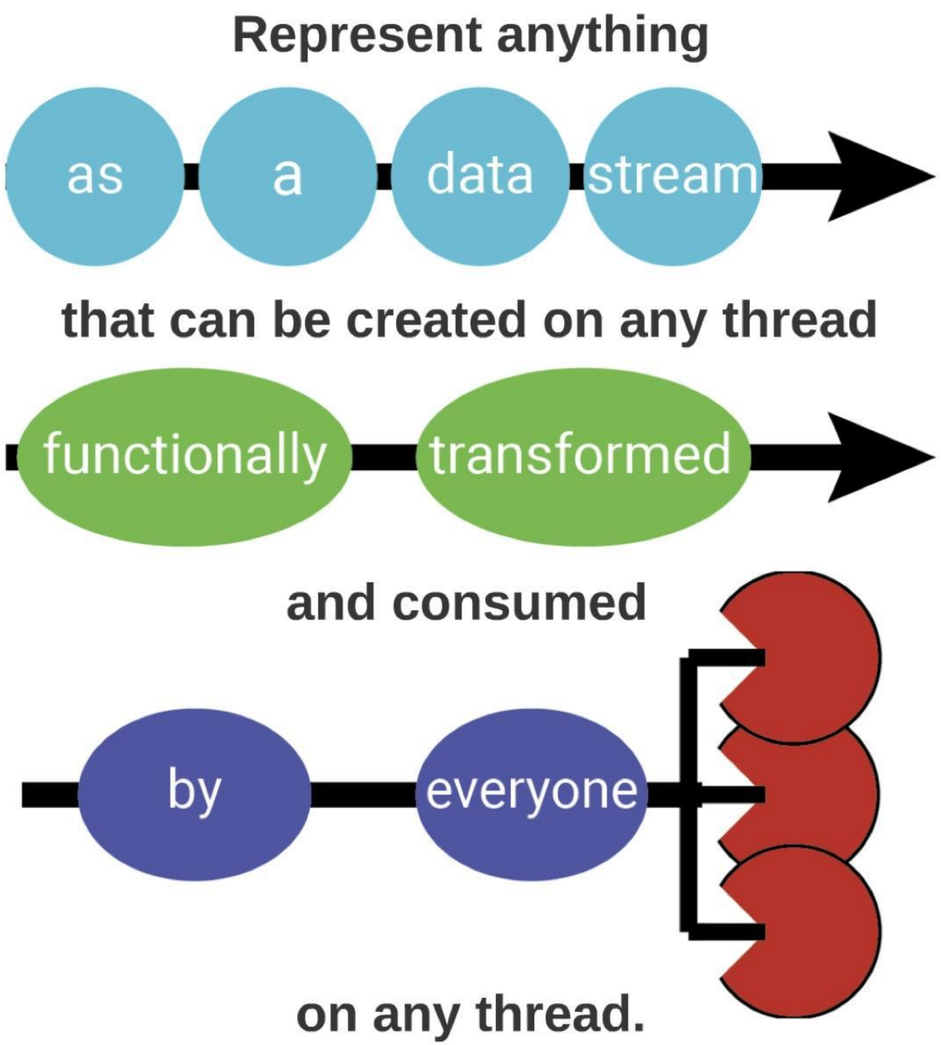
RXJAVA IS A JAVA VM IMPLEMENTATION OF REACTIVE EXTENSIONS: A LIBRARY FOR COMPOSING ASYNCHRONOUS AND EVENT-BASED PROGRAMS BY USING OBSERVABLE SEQUENCES.

Основные паттерны

- OBSERVABLE
- OBSERVER (SUBSCRIBER)
- OPERATORS
- SUBSCRIPTION
- SCHEDULERS

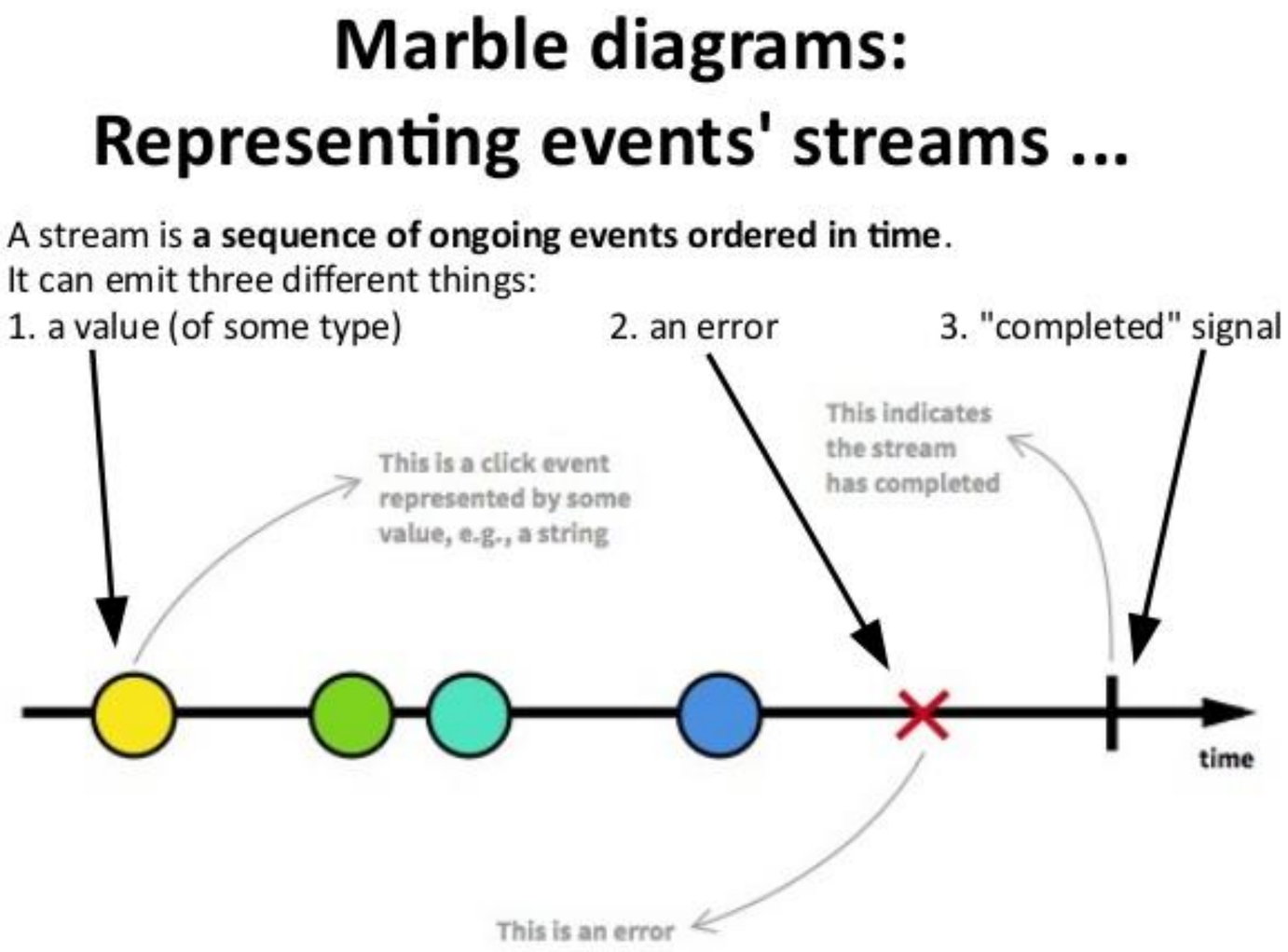


RXJAVA. SIMPLE. WHAT?

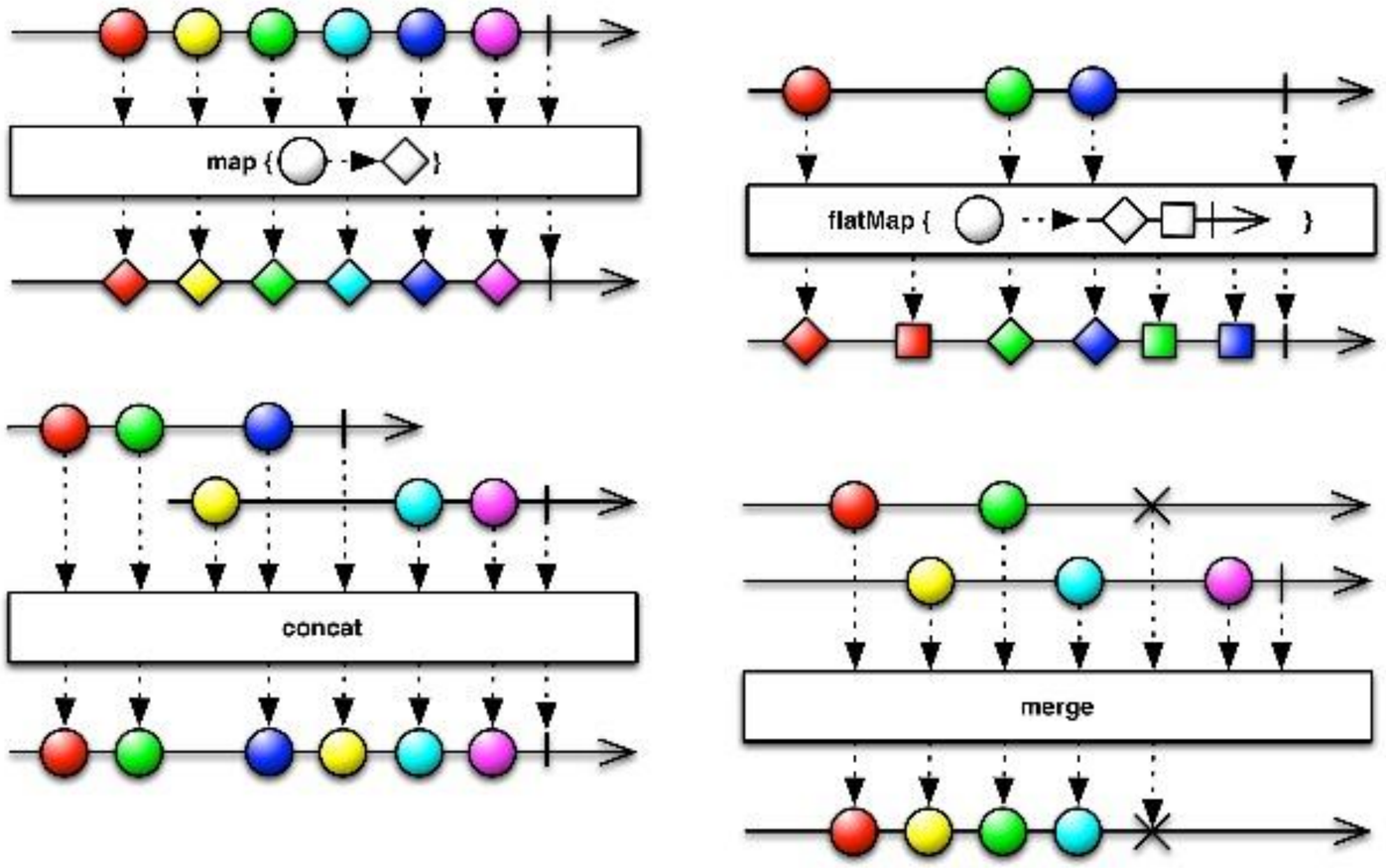


Prezi

RXJAVA. OBSERVABLE IS A STREAM.



RXJAVA. Операторы на MARBLE диаграммах



HTTPS://RXMARBLES.COM

RXJAVA SHEDULERS. RXANDROID

SCHEDULERS - особые операторы **RXJAVA**, предназначенные для выполнения операций над **OBSERVABLE** на разных потоках

Schedulers.io()
Schedulers.computation()
Schedulers.newThread()
Schedulers.single()
Schedulers.from(Executor executo
AndroidSchedulers.mainThread()

```
getItemsFromRemoteSource()
    .doOnNext { item -> Log.d(TAG, "Emitting item $item on thread ${currentThread().name}") }
    .subscribeOn(Schedulers.io())
    .observeOn(AndroidSchedulers.mainThread())
    .doOnNext { item -> Log.d(TAG, "Consuming item $item on thread ${currentThread().name}") }
    .subscribe { item -> showItem(item) }
```

RXANDROID – библиотека для **RXJAVA**, реализующая **ANDROIDSCHEDULER** для выполнения задач на основном потоке андроид приложения

HTTPS://GITHUB.COM/REACTIVEX/RXANDROID

RXJAVA и жи

```
class SampleActivity: AppCompatActivity() {

    private val presenter: SamplePresenter = SamplePresenter()

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        presenter.observeModel()
            .subscribeOn(Schedulers.io())
            .observeOn(AndroidSchedulers.main())
            .subscribe(
                { model -> bindModel(model) },
                { th -> bindError(th) }
            )
    }

    private fun bindModel(model: Model) { ... }

    private fun bindError(th: Throwable) { ... }

}
```

Проблемы:

- Утечка SAMPLEACTIVITY при повороте экрана
- Пересоздание презентера при повороте экрана

RXJAVA и жизненный цикл ACTIVITY/FRAGMENT

```
class SampleActivity: AppCompatActivity() {

    @Inject
    lateinit var presenter: SamplePresenter

    private var modelSubscription: Subscription? = null

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState, persistentState)
        ComponentManager.mainComponent.inject(this)
    }

    override fun onStart() {
        super.onStart()
        modelSubscription = presenter.observeModel()
            .subscribeOn(Schedulers.io())
            .observeOn(AndroidSchedulers.main())
            .subscribe(
                { model -> bindModel(model) },
                { th -> bindError(th) }
            )
    }

    override fun onStop() {
        super.onStop()
        modelSubscription?.unsubscribe()
    }
}
```

- Иньекция презентера и отписка на ONSTOP() решают проблему

RXJAVA и жизненный цикл ACTIVITY/FRAGMENT. RXLIFECYRCLE

```
class SampleActivity: RxActivity() {

    @Inject
    lateinit var presenter: SamplePresenter

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        ComponentManager.mainComponent.inject(this)
        presenter.observeModel()
            .subscribeOn(Schedulers.io())
            .compose(bindToLifecycle())
            .subscribe(
                { userInfo -> bindUserInfo() },
                { th -> bindError() }
            )
    }
}
```

RXLIFECYCLE позволяет автоматически завершать **RX** подписки по событиям жизненного цикла компонентов андроида

HTTPS://GITHUB.COM/TRELLO/RXLIFECYCLE

RXJAVA. Типичные варианты применения

- Поход в сеть
- Маппинг запросов
- Горячие подписки на обновление модели
- Временные отсчеты (DEBOUNCE, таймер)
- Ретраи запроса (поллинг)

...

RXJAVA и RETROFIT. Реактивный поход в сеть

RETROFIT — TYPE-SAFE HTTP CLIENT FOR ANDROID AND JAVA BY SQUARE, INC.

HTTPS://SQUARE.GITHUB.IO/RETROFIT/

```
interface ScalaApi {

    @GET("magazine/articles/snippets")
    fun getJournalArticles(
            @Query("category") articleCategory: Array<Category>,
            @Query("mark") mark: String,
            @Query("model") model: String,
            @Query("super_gen_id") superGen: String?,
            @Query("page_size") pageSize: Int
    ): Single<NWJournalSnippetsResponse>

    @GET("user/offers/{category}")
    fun getUserOffers(
            @Path("category") category: String = "all",
            @Query("page") page: Int,
            @Query("sort") sort: String?,
            @Query("page_size") pageSize: Int,
            @Query("with_daily_counters") withDailyViews: Boolean = false,
            @Query("moto_category") motoCategory: List<String>? = null,
            @Query("truck_category") truckCategory: List<String>? = null,
            @Query("section") state: String? = null,
            @Query("status") status: String? = null,
            @Query("mark_model") markModel: String? = null,
            @Query("price_from") priceFrom: Int? = null
```

MVVM

```
implementation "com.squareup.retrofit2:retrofit:$retrofit"
implementation "com.squareup.retrofit2:converter-gson:$retrofit"
implementation("com.squareup.retrofit2:converter-protobuf:$retrofit") {
    transitive = false;
}
implementation "com.squareup.retrofit2:adapter-rxjava:$retrofit"
```

RXJAVA

```
class OfferDetailsInteractor(
    private val offersRepository: IOffersRepository,
    private val userRepo: IUserOffersRepository,
    private val geoRepository: IGeoRepository,
    private val recentBadgesRepository: IRecentBadgesRepository
) : IOfferDetailsInteractor {

    override fun getOffer(
        category: String,
        offerId: String,
        isUserOffer: Boolean,
        rids: List<Int>?,
        geoRadius: Int?
    ): Single<Offer> = Single.zip(
        recentBadgesRepository.observeBadges().take(1).toSingle(),
        getOffer(isUserOffer, category, offerId, rids, geoRadius),
        { badges, offer -> offer.enrichWithRecentBadges(badges) }
    )
        .doOnSuccess { cacheOffer(it) }

    private fun getOffer(
        isUserOffer: Boolean,
        category: String,
        offerId: String,
        rids: List<Int>?,
        geoRadius: Int?
    ): Single<Offer> = when {
        isUserOffer -> offersRepository.getUserOffer(category, offerId)
        else -> offersRepository.getOffer(category, offerId, rids, geoRadius)
    }
}
```

RXJAVA. Горячие подписки на обновление модели

```
class FavoriteOfferInteractor(...): IFavoriteInteractor<Offer> {

    private val eventsSubj = PublishSubject.create<FavoriteSwitch<Offer>>().toSerialized()

    override fun switchFavorite(favorite: Offer): Completable = when {
        favoriteRepo.idsCache.contains(favorite.id) -> removeFavorite(favorite)
        else -> addFavorite(favorite)
    }

    override fun favoriteSwitchEvents(): Observable<FavoriteSwitch<Offer>> = eventsSubj

}
```

```
▼ Found usages 6 usages
  ▼ Production 6 usages
    ▼ FavoriteActionsController.kt 1 usage
        59 favoriteInteractor.favoriteSwitchEvents().silentLifeCycle {
    ▼ FavoriteOfferInteractor.kt 1 usage
        67 favoriteSwitchEvents().filter { offer -> predicate(offer.item) }
    ▼ FeedPresenter.kt 1 usage
        137 observeChangeEvents(favoritesInteractor.favoriteSwitchEvents()) { item, event ->
    ▼ FullScreenPhotoPresenter.kt 2 usages
        52 favoriteInteractor.favoriteSwitchEvents().silentLifeCycle {
        73 favoriteInteractor.favoriteSwitchEvents().silentLifeCycle { isFav ->
    ▼ OfferPostFeedLoader.kt 1 usage
        56 favoritesInteractor.favoriteSwitchEvents()
```

Яндекс

Спасибо

Даниэл Сергеев