



# Scala on Android

The current state of affairs

whoami

⇒ CTO & Co-Founder at 47 Degrees, a global consulting agency & Typesafe Consulting Partner.



@raulraja

@47deg

<http://47deg.com/blog>

# Build Tools



# SBT ⇒ Android SDK Plugin

<https://github.com/pfn/android-sdk-plugin>

# SBT ⇒ Android SDK Plugin

- Supports all Android SDK tasks
  - dex
  - *typedResourcesGenerator*
  - *proguard*
  - buildConfigGenerator
  - (+ 20... more)

# SBT ⇒ Android SDK Plugin ⇒ typedResourcesGenerator

```
object TR {  
    val title = TypedResource[TextView](R.id.title)  
  
    object layout {  
        val abc_screen_toolbar = TypedLayout[ActionBarOverlayLayout](R.layout.abc_screen_toolbar)  
    }  
}  
  
class MyActivity extends TypedActivity {  
  
    val titleTextView = findView(title) //titleTextView inferred as TextView, no casting needed  
}
```

SBT  $\Rightarrow$  Android SDK Plugin  $\Rightarrow$  *proguard*

# Size Matters

Scala byte code size reduced ~ (2.8M)

## SBT ⇒ Android SDK Plugin

- <https://github.com/pfn/android-sdk-plugin>
- Active
- Fast (incremental compilation and proguard caching)
- Proguard + MultiDexApplication integration (*Circumvents 65K method limit*)
- Supports AAR, JAR and APK artifact types

# IDE Support

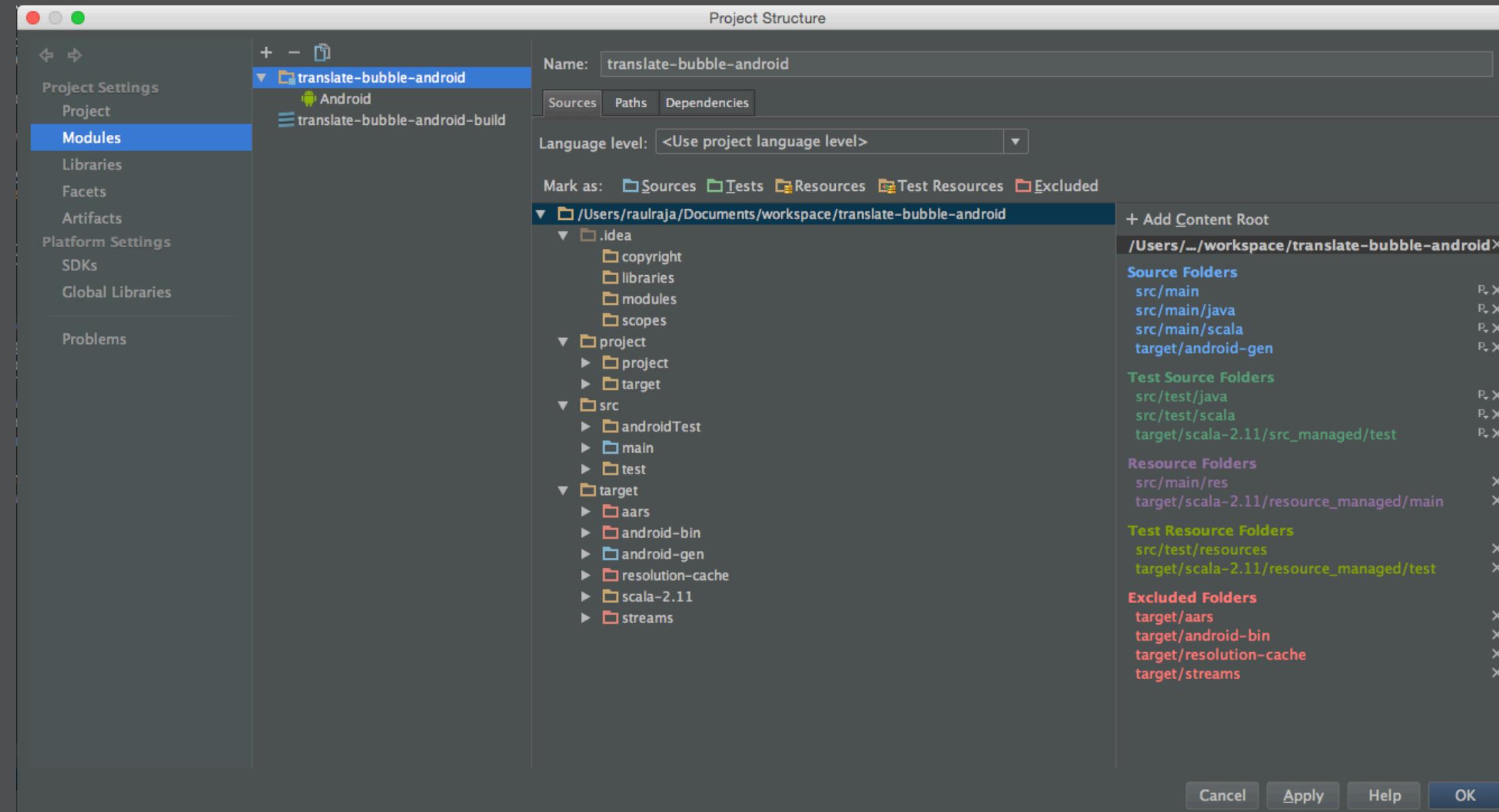


(Scala on Android) ⇒ The current state of affairs

# IntelliJ IDEA ⇒ *Syntax Highlighting*

```
17 package com.fortysevendeg.translatebubble.modules.notifications.impl
18
19 import android.app.{Notification, NotificationManager, PendingIntent}
20 import android.content.{Context, Intent}
21 import android.support.v4.app.NotificationCompat
22 import com.fortysevendeg.macroid.extras.AppContextProvider
23 import com.fortysevendeg.translatebubble.R
24 import com.fortysevendeg.translatebubble.modules.notifications._
25 import com.fortysevendeg.translatebubble.modules.persistent.PersistentServicesComponent
26 import com.fortysevendeg.translatebubble.service.Service
27 import com.fortysevendeg.translatebubble.ui.preferences.MainActivity
28
29 import scala.concurrent.ExecutionContext.Implicits.global
30 import scala.concurrent.Future
31
32 trait NotificationsServicesComponentImpl
33 extends NotificationsServicesComponent {
34
35   self : PersistentServicesComponent with AppContextProvider =>
36
37 lazy val notificationsServices = new NotificationsServicesImpl
38
39 class NotificationsServicesImpl
40 extends NotificationsServices {
```

# IntelliJ IDEA $\Rightarrow$ *Android + Scala + SBT*



(Scala on Android)  $\Rightarrow$  The current state of affairs

# IntelliJ IDEA

- Active
- Syntax Highlighting
- Code assistance
- *Android Studio* : Based on IntelliJ
  - > Google says: If you have been using Eclipse with ADT, be aware that Android Studio is now the official IDE for Android

# Libraries



Macroid



S

# Libraries ⇒ Scaloid ⇒ *Simplifying the Android SDK*

```
//plain vanilla scala
val button = new Button(context)
button.setText("Greet")
button.setOnClickListener(new OnClickListener() {
    def onClick(v: View) {
        Toast.makeText(context, "Hello!", Toast.LENGTH_SHORT).show()
    }
})
layout.addView(button)
```

# Libraries ⇒ Scaloid ⇒ *Simplifying the Android SDK*

```
//with Scaloid  
SButton("Greet", toast("Hello!"))
```

# Libraries ⇒ Scaloid ⇒ XML-less layouts

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical" android:layout_width="match_parent"
    android:layout_height="wrap_content" android:padding="20dip">
    <TextView android:layout_width="match_parent"
        android:layout_height="wrap_content" android:text="Sign in"
        android:layout_marginBottom="25dip" android:textSize="24.5sp"/>
    <TextView android:layout_width="match_parent"
        android:layout_height="wrap_content" android:text="ID"/>
    <EditText android:layout_width="match_parent"
        android:layout_height="wrap_content" android:id="@+id/userId"/>
    <TextView android:layout_width="match_parent"
        android:layout_height="wrap_content" android:text="Password"/>
    <EditText android:layout_width="match_parent"
        android:layout_height="wrap_content" android:id="@+id/password"
        android:inputType="textPassword"/>
    <Button android:layout_width="match_parent"
        android:layout_height="wrap_content" android:id="@+id/signin"
        android:text="Sign in"/>
</LinearLayout>
```

# Libraries ⇒ Scaloid ⇒ XML-less layouts

```
//with scaloid
new SVerticalLayout {
    STextView("Sign in").textSize(24.5 sp).<<.marginBottom(25 dip).>>
    STextView("ID")
    SEditText()
    STextView("Password")
    SEditText() inputType TEXT_PASSWORD
    SButton("Sign in")
}.padding(20 dip)
```

# Libraries ⇒ Scaloid ⇒ *Futures & Async*

```
//plain scala
new AsyncTask[String, Void, String] {
    def doInBackground(params: Array[String]) = {
        doAJobTakeSomeTime(params)
    }

    override def onPostExecute(result: String) {
        alert("Done!", result)
    }
}.execute("param")
```

# Libraries ⇒ Scaloid ⇒ *Futures & Async*

```
//scaloid  
Future {  
    val result = doAJobTakeSomeTime(params)  
    runOnUiThread(alert("Done!", result))  
}
```

# Libraries ⇒ Macroid ⇒ *Functional UI with Macros*

```
import macroid._  
import macroid.FullDsl._  
  
class GreetingActivity extends Activity with Contexts[Activity] {  
    override def onCreate(savedInstanceState: Bundle) = {  
        super.onCreate(savedInstanceState)  
  
        // the layout goes here  
        setContentView {  
            getUi {  
                l[LinearLayout](  
                    w[Button],  
                    w[TextView]  
                )  
            }  
        }  
    }  
}
```

# Libraries ⇒ Macroid ⇒ *UI composition*

```
// ActivityContext is an Android Context obtained from an Activity
import macroid.ActivityContext
...
def layout1(implicit ctx: ActivityContext) =
  l[LinearLayout](
    w[TextView]
  )

def layout2(implicit ctx: ActivityContext) =
  l[FrameLayout](
    w[ProgressBar]
  )

def layout3(implicit ctx: ActivityContext) =
  l[FrameLayout](
    layout1,
    layout2
  )
...
```

# Libraries ⇒ Macroid ⇒ *Tweaks*

```
l[LinearLayout]()

    // set button caption
    w[Button] <~ text("Click me"),

    // set text and hide for the time being
    w[TextView] <~ text("Hello!") <~ hide

    // set layout orientation
) <~ vertical
```

# Libraries ⇒ Macroid ⇒ *Tweaks Composition*

```
// AppContext is an Android Context obtained from getApplicationContext
import macroid.AppContext
// More tweaks
import macroid.contrib.TextTweaks

def greeting(greeting: String)(implicit appCtx: AppContext) =
    TextTweaks.large +
    text(greeting) +
    hide
```

# Libraries ⇒ Macroid ⇒ *Events*

```
button <~ On.click {  
    ...  
}
```

# Libraries ⇒ Macroid ⇒ Slots

```
// create a slot
var greeting = slot[TextView]

l[LinearLayout](
    w[TextView] <~
        // wire the view to the slot
        wire(greeting) <~
            OurTweaks.greeting("Hello!"),
    w[Button] <~
        text("Click me") <~
        On.click {
            // use the slot elsewhere
            greeting <~ show
        }
) <~ vertical
```

# Libraries ⇒ Macroid ⇒ *Contexts*

```
class MyActivity extends Activity with Contexts[Activity] {  
    // implicit access to ApplicationContext & ActivityContext stored as a weak reference  
}
```

```
class MyFragment extends Fragment with Contexts[Fragment] {  
    // implicit access to ApplicationContext & ActivityContext stored as a weak reference  
}
```

# Libraries ⇒ Macroid ⇒ *Contexts*

```
class MyActivity extends Activity with Contexts[Activity] {  
    // implicit access to ApplicationContext & ActivityContext stored as a weak reference  
}
```

```
class MyFragment extends Fragment with Contexts[Fragment] {  
    // implicit access to ApplicationContext & ActivityContext stored as a weak reference  
}
```

No need to explicitly pass the context around!

# Libraries ⇒ Macroid ⇒ *Snails*

```
val focusLoudly = Snail[View] { view =>  
    view.setFocus()  
    playSound  
}
```

# Libraries ⇒ Macroid ⇒ *Snails composition*

```
val wink = fadeIn ++ fadeOut
```

**Libraries ⇒ Macroid ⇒ *Snails & Tweaks combined***

```
editText <~ text("foo") <~~ fadeIn <~ enable
```

# Libraries ⇒ Macroid ⇒ *Easy workflows*

```
(myProgressBar <~~ fadeOut(400)) ~~  
(myTextView <~~ blink) ~~  
(myOtherTextView <~ text("Scala at the Sea!"))
```

# Libraries ⇒ Macroid ⇒ *Transformers*

```
linearLayout <~ Transformer {  
    case t: TextView ⇒ t <~ text("foo")  
    case i: ImageView ⇒ i <~ hide  
}
```

# Libraries ⇒ Macroid ⇒ *Adaptive (Media queries)*

```
object OurTweaks {  
    def orient(implicit appCtx: AppContext) =  
        landscape ? horizontal | vertical  
}  
  
...  
// in layout  
l[LinearLayout] (  
    ...  
) <~ OurTweaks.orient
```

# Libraries $\Rightarrow$ Macroid $\Rightarrow$ *Viewable*

Brings the power of Typeclasses to UI composition.

How to display *A* using *W*.

```
libraryDependencies += aar("org.macroid" %% "macroid-viewable" % "2.0.0-M3")
```

```
trait Viewable[A, +W <: View]
```

# Libraries ⇒ Macroid ⇒ *Viewable*

```
import macroid.viewable.Viewable

case class User(name: String)

def userViewable(
    implicit ctx: ActivityContext,
    appCtx: AppContext): Viewable[User, TextView] =
    Viewable[User] { user =>
        w[TextView] <~ TextTweaks.large <~ text(user.name)
    }
```

# Libraries ⇒ Macroid ⇒ *Akka Fragments*

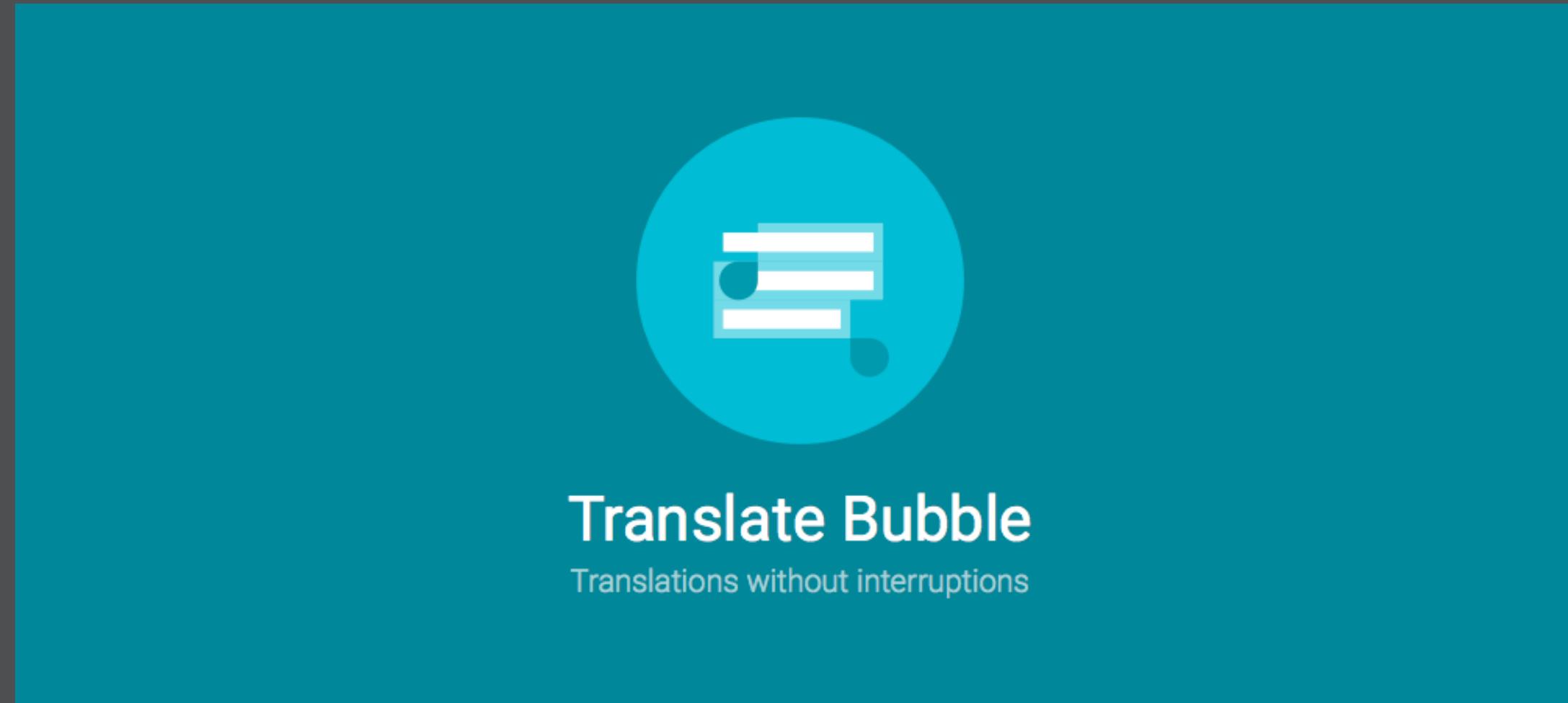
Handling Fragment events & messaging with Actors.

```
class MyActor extends FragmentActor[MyFragment] {  
    def receive = receiveUi andThen {  
        case MyMessage(x) => ...  
  
        case MyOtherMessage => withUi(fragment => Ui {  
            // do some cool ui stuff here  
        })  
        case FragmentActor.AttachUi(_) => ...  
        case FragmentActor.DetachUi(_) => ...  
    }  
}
```

# OS Apps $\Rightarrow$ *Translate Bubble*

<https://play.google.com/store/apps/details?id=com.fortysevendeg.translatebubble>

<https://github.com/47deg/translate-bubble-android>



OS Apps ⇒ *Scala Days Official App*

<https://play.google.com/store/apps/details?id=com.fortysevendeg.android.scaladays>

<https://github.com/47deg/scala-days-android>

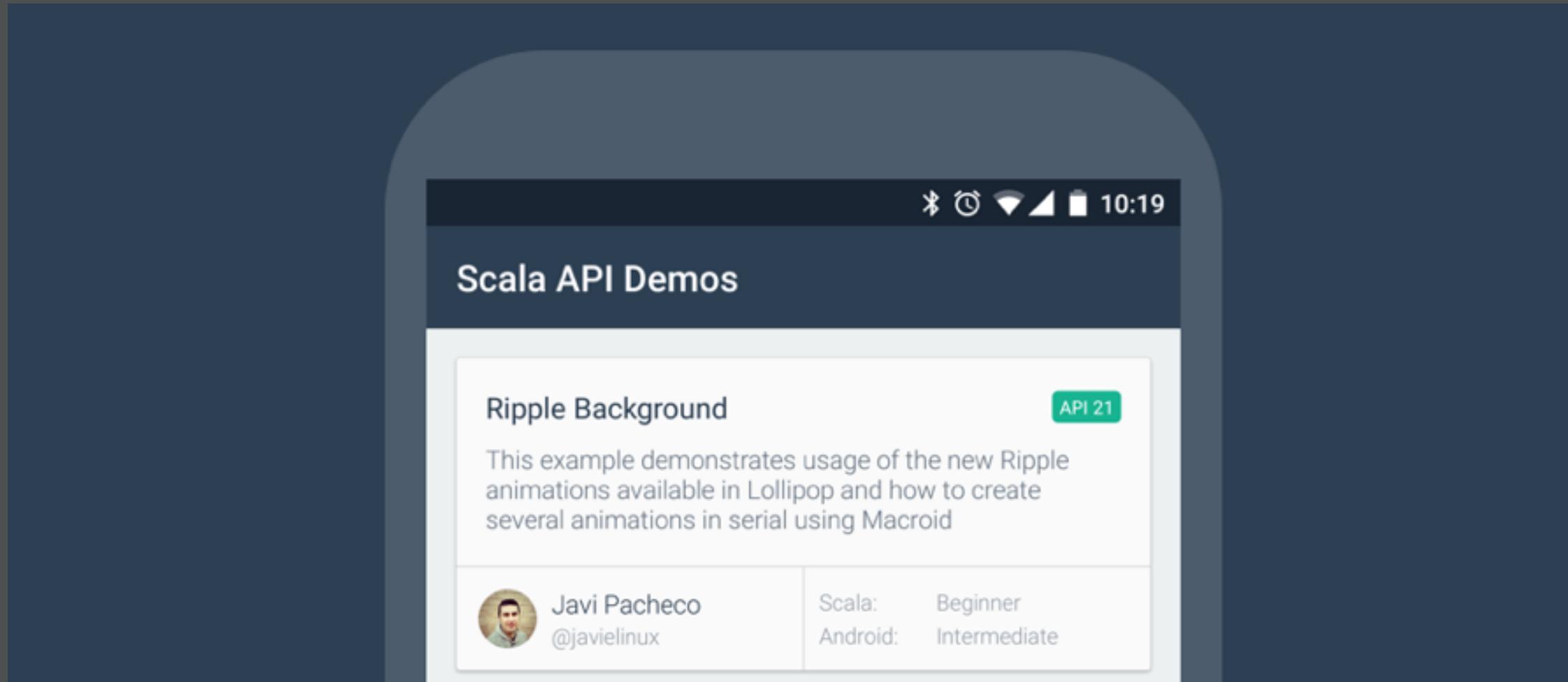
(source code available on March 15th)



# OS Apps ⇒ *Scala API Demos*

<https://play.google.com/store/apps/details?id=com.fortysevendeg.scala.android>

<https://github.com/47deg/scala-android>



# Thank you

@raulraja

@47deg

raul at 47deg.com

<http://47deg.com/blog>

<https://speakerdeck.com/raulraja/scala-on-android-the-current-state-of-affairs>

<https://github.com/47deg/scala-on-android-deck>