+

Scala + Android

Konrad Malawski / @ktosopl
GeeCON / GDG / PJUG / KSUG / SCKRK
JFokus **06.02.2013**

softwaremill

polish java user group

geecon

SSEPP STOWARZYSZENIE SOFTWARE ENGINEERING PROFESSIONALS POLSKA

compendium centrum edukacyjne

Konrad Malawski

twitter: ktosopl

[sckrk]

GDG Kraków

KRAKOW SCALA USER GROUP

This will be...

This will **Not** be...

This will **Not** be...

This will **Not** be...

*a **Java** rant,*

about app design.

This will be about...

This will be about...

This will be about...

shifting cognitive load

writing ***less code***

fokus *on* ***goals****, not means*

Scala is

Scala is

Scala is

Simple,
but Hard.

Scala is Simple.

ですので、こんなコードはコンパイル通ります。

```
val どんな空気よ_? = (なぜか変換できない : List[String]) ⇒
  for( ふいんき ← なぜか変換できない ) {
    println( ふいんき )
  }
どんな空気よ_?(List( "よい" ,"悪い"))
```

また、"→(Unicode \u2192)"は、scala.predef.ArrowAssocの別名です。scalaでは、"key" -> "value"でTuple2("key","value")を生成できるのですが、"key" → "value"でも同様にTupleができます。

なので、Mapの初期値を設定するときには、こんな風に書けます。

```
val m = Map( "日本語で書いても読みにくくね?" → "それをいっちゃあおしまいよ" ,
             "誰得?" → "おまえが言うな"  )
```

実際どうなるの?

で、まぁBefore→Afterでこんな感じです。
かえって読みにくくね?って質問は却下です。

Scala is Hard.

```
object Param {

  implicit def pToT[A[_], B[_]](f: (p: Param[A]) => B[p.T]): A~>B = new (A ~> B) {
    def apply[s](a: A[s]): B[s] = {
      val v: Param[A] { type T = s} = new Param[A] {
        type T = s
        def in = a
      }
      f(v)
    }
  }

}
```

http://apocalisp.wordpress.com/2010/10/26/

type-level-programming

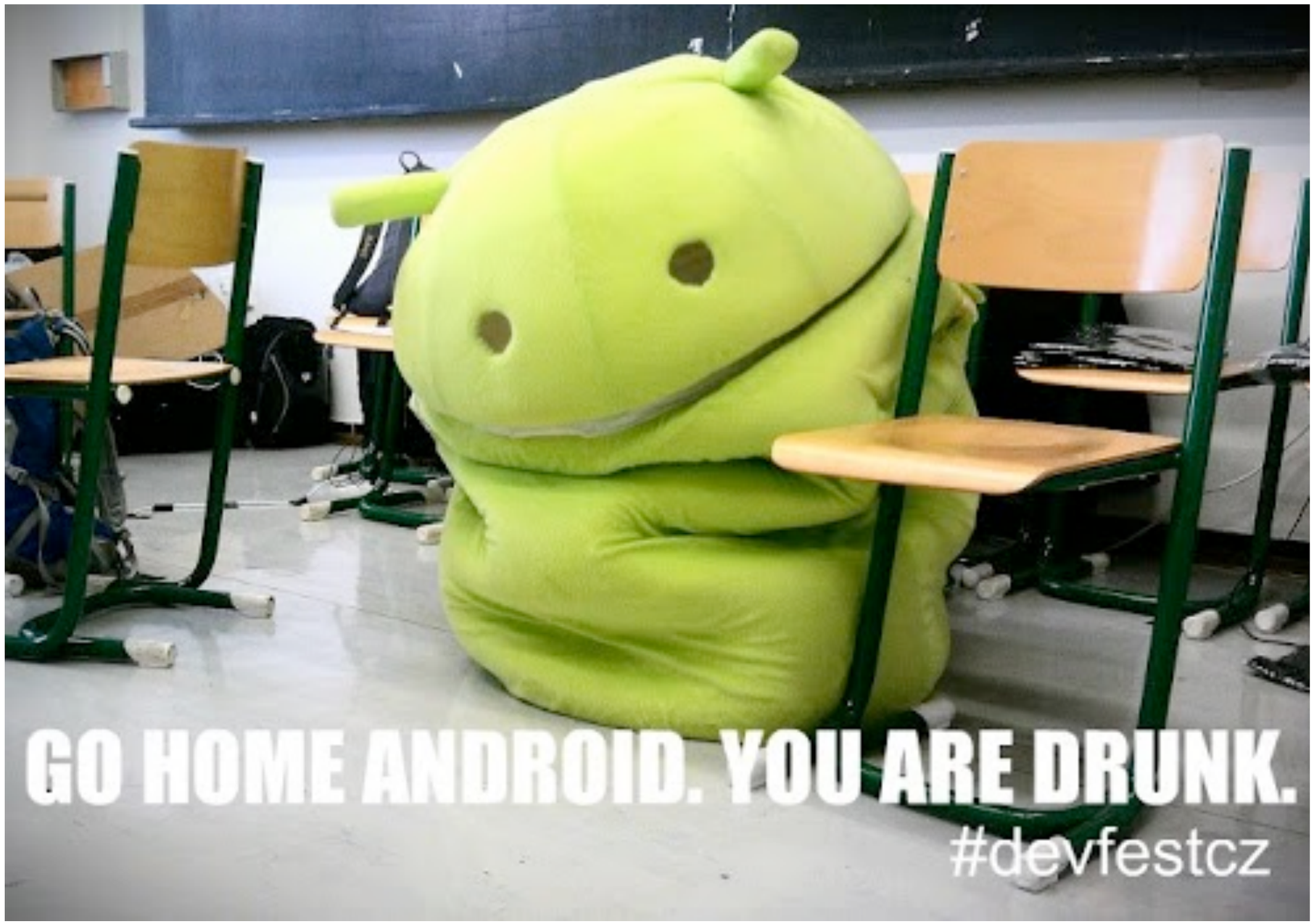
-in-scala-part-7-natural-transformation%C2%A0literals/

Scala is

A **Functional**,
Object Oriented,
Statically Typed,
Scalable,
Language...

...running on the **JVM** and **DVM**!

Android

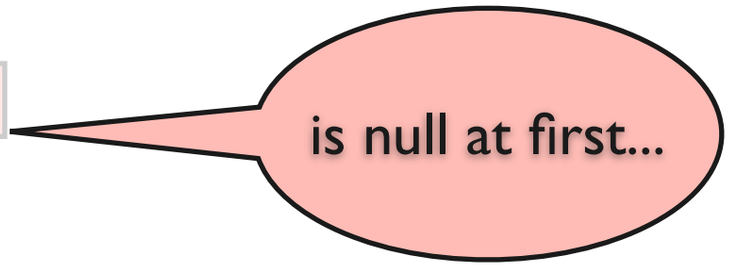


findViewById

```
public class MyActivity extends Activity {

  ListView comments;
  Button newComment;

  @Override
  void onCreate(Bundle bundle) {
    super.onCreate(bundle);
    comments = (ListView) findViewById(R.id.comments);
    newComment = (Button) findViewById(R.id.new_comment);
    // ...
  }
}
```

findViewById

```
public class MyActivity extends Activity {

  ListView comments;
  Button newComment;

  @Override
  void onCreate(Bundle bundle) {
    super.onCreate(bundle);
    comments = (ListView) findViewById(R.id.comments);
    newComment = (Button) findViewById(R.id.new_comment);
    // ...
  }
}
```



findViewById

```
public class MyActivity extends Activity {

  ListView comments;
  Button newComment;

  @Override
  void onCreate(Bundle bundle) {
    super.onCreate(bundle);
    comments = (ListView) findViewById(R.id.comments);
    newComment = (Button) findViewById(R.id.new_comment);
    // ...
  }
}
```

initialized in method...

findViewById

```
public class MyActivity extends Activity {

  ListView comments;
  Button newComment;

  @Override
  void onCreate(Bundle bundle) {
    super.onCreate(bundle);
    comments = (ListView) findViewById(R.id.comments);
    newComment = (Button) findViewById(R.id.new_comment);
    // ...
  }
}
```

Explicit casting!

Robo Guice

So many words...

```
public class MakeANoteActivity
    extends RoboActivity
    implements View.OnClickListener
{
    public static final String PHOTO_PATH = "picture_path";
    public static final String PHOTO_ATTACHED = "photo_attached";

    @Inject SessionManager sessionManager;
    @Inject LocationHelper locationHelper;
    @Inject PhotoHelper photoHelper;
    @Inject Application context;

    @Inject CalibrationHelper calibrationHelper;

    @InjectView(R.id.attach_photo) Button attachPhoto;
    @InjectView(R.id.note_text) EditText noteText;
    @InjectView(R.id.save_button) Button save;
    @InjectView(R.id.date) TextView dateText;
    @InjectView(R.id.share) Button share;

    @InjectResource(R.string.im_aircasting) String imAircasting;
    @InjectResource(R.string.share_with) String shareWith;

    // ...
}
```

Robo **Guice**

```
public class MakeANoteActivity
    extends RoboActivity
    implements View.OnClickListener
{
     public static final String PHOTO_PATH = "picture_path";
     public static final String PHOTO_ATTACHED = "photo_attached";

    @Inject SessionManager sessionManager;
    @Inject LocationHelper locationHelper;
    @Inject PhotoHelper photoHelper;
    @Inject Application context;

    @Inject CalibrationHelper calibrationHelper;

    @InjectView(R.id.attach_photo) Button attachPhoto;
    @InjectView(R.id.note_text) EditText noteText;
    @InjectView(R.id.save_button) Button save;
    @InjectView(R.id.date) TextView dateText;
    @InjectView(R.id.share) Button share;

    @InjectResource(R.string.im_aircasting) String imAircasting;
    @InjectResource(R.string.share_with) String shareWith;

    // ...
}
```

Guice is eager.
Startup can take a few seconds!

Robo **Guice**

```
public class MakeANoteActivity
    extends RoboActivity
    implements View.OnClickListener
{
     public static final String PHOTO_PATH = "picture_path";
     public static final String PHOTO_ATTACHED = "photo_attached";

     @Inject SessionManager sessionManager;
     @Inject LocationHelper locationHelper;
     @Inject PhotoHelper photoHelper;
     @Inject Application context;

     @Inject CalibrationHelper calibrationHelper;

     @InjectView(R.id.attach_photo) Button attachPhoto;
     @InjectView(R.id.note_text) EditText noteText;
     @InjectView(R.id.save_button) Button save;
     @InjectView(R.id.date) TextView dateText;
     @InjectView(R.id.share) Button share;

     @InjectResource(R.string.im_aircasting) String imAircasting;
     @InjectResource(R.string.share_with) String shareWith;

     // ...
}
```

Type changes in XML,
but not here...
ClassCastException!

Stairway to **Scala**

Collections

(the gateway drug)

Collections

POJO

```java
public class Person {
  private final String name;
  private final String nick;

    public Person(String name, String nick) {
        this.name = name;
        this.nick = nick;
    }

    public String getName() {
        return name;
    }

    public String getNick() {
        return nick;
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (!(o instanceof Person)) return false;

        Person person = (Person) o;

        if (name != null ? !name.equals(person.name) : person.name != null) return false;
        if (nick != null ? !nick.equals(person.nick) : person.nick != null) return false;

        return true;
    }

    @Override
    public int hashCode() {
        int result = name != null ? name.hashCode() : 0;
        result = 31 * result + (surname != null ? surname.hashCode() : 0);
        return result;
    }
}
```

POJO*

*Actually just a **Case Class**

```
public class Person {
  private final String name;
  private final String nick;

    public Person(String name, String nick) {
        this.name = name;
        this.nick = nick;
    }

    public String getName() {
        return name;
    }

    public String getNick() {
        return nick;
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (!(o instanceof Person)) return false;

        Person person = (Person) o;

        if (name != null ? !name.equals(person.name) : person.name != null) return false;
        if (nick != null ? !nick.equals(person.nick) : person.nick != null) return false;

        return true;
    }

    @Override
    public int hashCode() {
        int result = name != null ? name.hashCode() : 0;
        result = 31 * result + (surname != null ? surname.hashCode() : 0);
        return result;
    }
}
```

```
case class Person(name: String, nick: String)
```

POSO*

*Actually just a **Case Class**

val name: String

```
case class Person(name: String, nick: String)
```

Case Classes

```
val person = Person("Bruce Wayne", "Manbat") // whoops!
val fixed  = person.copy(nick = "Batman")
     case class Person(name: String, nick: String)

val self   = Person(nick = "Ktoso" , name = "Konrad")

val Person(name, hero) = person
val hello = "Hello" + name + " " + surname


fixed.toString should equal ("Person(Bruce Wayne, Batman)")
```

Still thinking assertThat().isEqualTo() is clean?

Scala Collections at Work

Given a **list of People**,
return all the **Hero names**.

```
def heroNames(people: List[Person]): List[String] =
    people filter { _.isHero } map { _.name }
```

Back to **Android**

Another look at the Java code

```
public class MakeANoteActivity extends RoboActivity
    implements View.OnClickListener

{
  public static final String PHOTO_PATH = "picture_path";
  public static final String PHOTO_ATTACHED = "photo_attached";

  @Inject SessionManager sessionManager;
  @Inject LocationHelper locationHelper;
  @Inject PhotoHelper photoHelper;
  @Inject Application context;

  @Inject CalibrationHelper calibrationHelper;

  @InjectView(R.id.attach_photo) Button attachPhoto;
  @InjectView(R.id.note_text) EditText noteText;
  @InjectView(R.id.save_button) Button save;
  @InjectView(R.id.date) TextView dateText;
  @InjectView(R.id.share) Button share;

  @InjectResource(R.string.im_aircasting) String imAircasting;
  @InjectResource(R.string.share_with) String shareWith;

  // ...
}
```

What if... **Scala**?

```
public class MakeANoteActivity extends ScalaRoboActivity
    implements View.OnClickListener
    with    PhotoOperations with CalibrationOperations
{
  public final String PHOTO_PATH = "picture_path";
  public final String PHOTO_ATTACHED = "photo_attached";

  @Inject SessionManager sessionManager;
  @Inject LocationHelper locationHelper;
  @Inject PhotoHelper photoHelper;
  @Inject Application context;

  @Inject CalibrationHelper calibrationHelper;
  @InjectView(R.id.attach_photo) ImageView attachPhoto;
  @InjectView(R.id.note_text) EditText noteText;
  @InjectView(R.id.save_button) Button saveButton;
  @InjectView(R.id.date) TextView dateText;
  @InjectView(R.id.share) Button share;

  @InjectResource(R.string.username) String username;
  @InjectResource(R.array.list_of_friends) List<String> friends;

  def toastConferenceName() = "KrakDroid".toast()
}
```

What if... Scala?

```
class MakeANoteActivity extends ScalaActivity
  with    View.OnClickListener
  with    PhotoOperations with CalibrationOperations
{
  val PicturePath   = "picture_path"
  val PhotoAttached = "photo_attached"

  lazy val sessionManager  = SessionManager.get
  lazy val locationManager = LocationManager.get

  lazy val attachPhoto = findView(TR.attach_photo)
  lazy val noteText    = findView(TR.note_text)
  lazy val save        = findView(TR.save_button)
  lazy val dateText    = findView(TR.date)
  lazy val share       = findView(TR.share)


  val username = findResource[String](R.string.my_username)
  val friends  = findResource[List[String]](R.string.friends)

  def toastConferenceName() = "KrakDroid".toast()
}
```

public is default.

What if... Scala?

```
class MakeANoteActivity extends ScalaActivity
   with    View.OnClickListener
   with    PhotoOperations with CalibrationOperations
{
  val PhotoPath     = "picture_path"
  val PhotoAttached = "photo_attached"

  lazy val sessionManager  = SessionManager.get
  lazy val locationManager = LocationManager.get
```

ScalaActivity gives us some magic methods

```
  lazy val attachPhoto = findView(TR.attach_photo)
  lazy val noteText    = findView(TR.note_text)
  lazy val save        = findView(TR.save_button)
  lazy val dateText    = findView(TR.date)
  lazy val share       = findView(TR.share)


  val username = findResource[String](R.string.my_username)
  val friends  = findResource[List[String]](R.string.friends)

  def toastConferenceName() = "KrakDroid".toast()
}
```

What if... Scala?

```
class MakeANoteActivity extends ScalaActivity
   with    View.OnClickListener
   with    PhotoOperations with CalibrationOperations
{
  val PhotoPath      = "picture_path"
  val PhotoAttached = "photo_attached"

  lazy val sessionManager  = SessionManager.get
  lazy val locationManager = LocationManager.get
```

It's so easy,
YOU can implement it!

ScalaActivity gives us some magic methods

```
  lazy val attachPhoto = findView(TR.attach_photo)
  lazy val noteText    = findView(TR.note_text)
  lazy val save        = findView(TR.save_button)
  lazy val dateText    = findView(TR.date)
  lazy val share       = findView(TR.share)


  val username = findResource[String](R.string.my_username)
  val friends  = findResource[List[String]](R.string.friends)

  def toastConferenceName() = "KrakDroid".toast()
}
```

What if... Scala?

```
class MakeANoteActivity extends ScalaActivity
    with     View.OnClickListener
    with     PhotoOperations with CalibrationOperations
{
  val PhotoPath      = "picture_path"
  val PhotoAttached = "photo_attached"

  lazy val sessionManager  = SessionManager.get
  lazy val locationManager = LocationManager.get
```

Instead of "**...Helper**", use **Traits**.

```
  lazy val attachPhoto = findView(TR.attach_photo)
  lazy val noteText    = findView(TR.note_text)
  lazy val save        = findView(TR.save_button)
  lazy val dateText    = findView(TR.date)
  lazy val share       = findView(TR.share)


  val username = findResource[String](R.string.my_username)
  val friends  = findResource[List[String]](R.string.friends)

  def toastConferenceName() = "KrakDroid".toast()
}
```

What if... Scala?

```
class MakeANoteActivity extends ScalaActivity
   with    View.OnClickListener
   with    PhotoOperations with CalibrationOperations
{
  val PhotoPath      = "picture_path"
  val PhotoAttached = "photo_attached"

  lazy val sessionManager  = SessionManager.get
  lazy val locationManager = LocationManager.get

  lazy val attachPhoto = findView(TR.attach_photo)
  lazy val noteText    = findView(TR.note_text)
  lazy val save        = findView(TR.save_button)
  lazy val dateText    = findView(TR.date)
  lazy val share       = findView(TR.share)


  val username = findResource[String](R.string.my_username)
  val friends  = findResource[List[String]](R.string.friends)

  def toastConferenceName() = "KrakDroid".toast()
}
```

Type Inference

What if... Scala?

```
class MakeANoteActivity extends ScalaActivity
   with    View.OnClickListener
   with    PhotoOperations with CalibrationOperations
{
  val PhotoPath     = "picture_path"
  val PhotoAttached = "photo_attached"

  lazy val sessionManager  = SessionManager.get
  lazy val locationManager = LocationManager.get
```

Who's missing here?

```
  lazy val attachPhoto = findView(TR.attach_photo)
  lazy val noteText    = findView(TR.note_text)
  lazy val save        = findView(TR.save_button)
  lazy val dateText    = findView(TR.date)
  lazy val share       = findView(TR.share)


  val username = findResource[String](R.string.my_username)
  val friends  = findResource[List[String]](R.string.friends)

  def toastConferenceName() = "KrakDroid".toast()
}
```

What if... Scala?

```
class MakeANoteActivity extends ScalaActivity
   with    View.OnClickListener
   with    PhotoOperations with CalibrationOperations
{
  val PhotoPath      = "picture_path"
  val PhotoAttached = "photo_attached"

  lazy val sessionManager  = SessionManager.get
  lazy val locationManager = LocationManager.get

  lazy val attachPhoto = findView(TR.attach_photo)
  lazy val noteText    = findView(TR.note_text)
  lazy val save        = findView(TR.save_button)
  lazy val dateText    = findView(TR.date)
  lazy val share       = findView(TR.share)


  val username = findResource[String](R.string.my_username)
  val friends  = findResource[List[String]](R.string.friends)

  def toastConferenceName() = "KrakDroid".toast()
}
```

lazy initialization

What if... Scala?

```
class MakeANoteActivity extends ScalaActivity
   with    View.OnClickListener
   with    PhotoOperations with CalibrationOperations
{
  val PhotoPath     = "picture_path"
  val PhotoAttached = "photo_attached"

  lazy val sessionManager  = SessionManager.get
  lazy val locationManager = LocationManager.get
```

Needs **Context**

```
  lazy val attachPhoto = findView(TR.attach_photo)
  lazy val noteText    = findView(TR.note_text)
  lazy val save        = findView(TR.save_button)
  lazy val dateText    = findView(TR.date)
  lazy val share       = findView(TR.share)
```

Implicit Parameters

```
  val username = findResource[String](R.string.my_username)
  val friends  = findResource[List[String]](R.string.friends)

  def toastConferenceName() = "KrakDroid".toast()
}
```

What if... Scala?

```
class MakeANoteActivity extends ScalaActivity
   with    View.OnClickListener
   with    PhotoOperations with CalibrationOperations
{
  val PhotoPath     = "picture_path"
  val PhotoAttached = "photo_attached"

  lazy val sessionManager  = SessionManager.get
  lazy val locationManager = LocationManager.get
```

What's the type here?!?

```
  lazy val attachPhoto = findView(TR.attach_photo)
  lazy val noteText    = findView(TR.note_text)
  lazy val save        = findView(TR.save_button)
  lazy val dateText    = findView(TR.date)
  lazy val share       = findView(TR.share)

  val username = findResource[String](R.string.my_username)
  val friends  = findResource[List[String]](R.string.friends)

  def toastConferenceName() = "KrakDroid".toast()
}
```

What if... Scala?

```
class MakeANoteActivity extends ScalaActivity
   with    View.OnClickListener
   with    PhotoOperations with CalibrationOperations
{
  val PhotoPath     = "picture_path"
  val PhotoAttached = "photo_attached"

  lazy val sessionManager  = SessionManager.get
  lazy val locationManager = LocationManager.get

  lazy val attachPhoto = findView(TR.attach_photo)
  lazy val noteText    = findView(TR.note_text)
  lazy val save        = findView(TR.save_button)
  lazy val dateText    = findView(TR.date)
  lazy val share       = findView(TR.share)

  val username = findResource[String](R.string.my_username)
  val friends  = findResource[List[String]](R.string.friends)

  def toastConferenceName() = "KrakDroid".toast()
}
```

What's the type here?!?

Typed **R**esource - "a better R"

What if... Scala?

```
class MakeANoteActivity extends ScalaActivity
   with    View.OnClickListener
   with    PhotoOperations with CalibrationOperations
{
  val PhotoPath     = "picture_path"
  val PhotoAttached = "photo_attached"

  lazy val sessionManager  = SessionManager.get
  lazy val locationManager = LocationManager.get

  lazy val attachPhoto = findView(TR.attach_photo)
  lazy val noteText    = findView(TR.note_text)
  lazy val save        = findView(TR.save_button)
  lazy val dateText    = findView(TR.date)
  lazy val share       = findView(TR.share)

  val username = findResource[String](R.string.username)
  val friends  = findResource[List[String]](R.string.friends)

  def toastConferenceName() = "KrakDroid".toast()
}
```

What's the type here?!?

Typed **R**esource - "a better R"

XML **Type** changes... **Type** here changes!

What if... Scala?

```
class MakeANoteActivity extends ScalaActivity
   with    View.OnClickListener
   with    PhotoOperations with CalibrationOperations
{
  val PhotoPath      = "picture_path"
  val PhotoAttached = "photo_attached"

  lazy val sessionManager  = SessionManager.get
  lazy val locationManager = LocationManager.get

  lazy val attachPhoto   = findView(TR.attach_photo)
  lazy val noteText      = findView(TR.note_text)
  lazy val save          = findView(TR.save_button)
  lazy val dateText      = findView(TR.date)
  lazy val share: Button = findView(TR.share)

  val username = findResource[String](R.string.username)
  val friends  = findResource[List[String]](R.string.friends)

  def toastConferenceName() = "KrakDroid".toast()
}
```

What's the type here?!?

Typed **R**esource - "a better R"

You **can** be **explicit** though!

What if... Scala?

```
class MakeANoteActivity extends ScalaActivity
   with    View.OnClickListener
   with    PhotoOperations with CalibrationOperations
{
  val PhotoPath     = "picture_path"
  val PhotoAttached = "photo_attached"

  lazy val sessionManager  = SessionManager.get
  lazy val locationManager = LocationManager.get

  lazy val attachPhoto = findView(TR.attach_photo)
  lazy val noteText    = findView(TR.note_text)
  lazy val save        = findView(TR.save_button)
  lazy val dateText    = findView(TR.date)
  lazy val share       = findView(TR.share)

  val username = findResource[String](R.string.my_username)
  val friends  = findResource[List[String]](R.string.friends)

  def toastConferenceName() = "KrakDroid".toast()
}
```

Like @InjectResource, but smarter

Manifests vs. Type Erasure

What if... Scala?

```
class MakeANoteActivity extends ScalaActivity
   with    View.OnClickListener
   with    PhotoOperations with CalibrationOperations
{
  val PhotoPath     = "picture_path"
  val PhotoAttached = "photo_attached"

  lazy val sessionManager  = SessionManager.get
  lazy val locationManager = LocationManager.get

  lazy val attachPhoto = findView(TR.attach_photo)
  lazy val noteText    = findView(TR.note_text)
  lazy val save        = findView(TR.save_button)
  lazy val dateText    = findView(TR.date)
  lazy val share       = findView(TR.share)

  val username = findResource[String](R.string.username)
  val friends  = findResource[List[String]](R.string.friends)

  def toastConferenceName() = "KrakDroid".toast()
}
```

Does **String** have a **toast()** method?

Implicit Conversion

Typed **R**esource

Typed **R**esource = Better **R**

```scala
class MakeANoteActivity extends ScalaActivity
   with    View.OnClickListener
   with    PhotoOperations with CalibrationOperations
{
  val PhotoPath     = "picture_path"
  val PhotoAttached = "photo_attached"

  lazy val sessionManager  = SessionManager.get
  lazy val locationManager = LocationManager.get
```

Typed Resource = Better R

```scala
  lazy val attachPhoto = findView(TR.attach_photo)
  lazy val noteText    = findView(TR.note_text)
  lazy val save        = findView(TR.save_button)
  lazy val dateText    = findView(TR.date)
  lazy val share       = findView(TR.share)

  val username = findResource[String](R.string.my_username)
  val friends  = findResource[List[String]](R.string.friends)

  def toastConferenceName() = "KrakDroid".toast()
}
```

Typed Resource

Typed Resource = Better R

```
<TableLayout android:id="@+id/login_table"
                    android:layout_gravity="center"
                    android:layout_height="wrap_content"
                    android:layout_width="wrap_content">
```

```
case class TypedResource[T](id: Int)
case class TypedLayout(id: Int)
```

Wrappers

```
object TR {
  val login_table = TypedResource[TableLayout](R.id.login_table)

  val workspaces = TypedResource[ExpandableListView](R.id.workspaces)

  val subtask_txt = TypedResource[TextView](R.id.subtask_txt)
 // ...

}
```

Typed Resource = Better R

```
<TableLayout android:id="@+id/login_table"
             android:layout_gravity="center"
             android:layout_height="wrap_content"
             android:layout_width="wrap_content">
```

```
case class TypedResource[T](id: Int)
case class TypedLayout(id: Int)
```

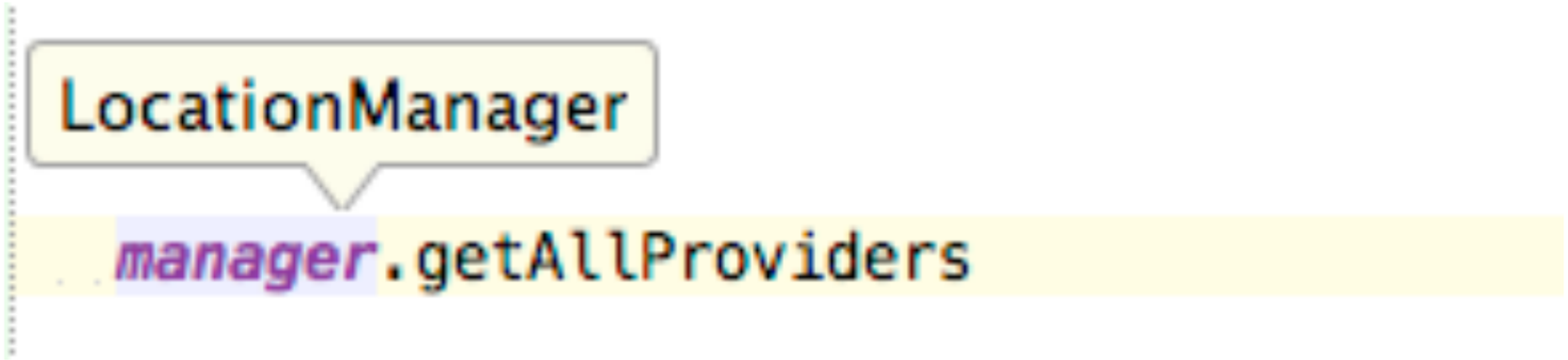
```
object TR {
  val login_table = TypedResource[TableLayout](R.id.login_table)

  val workspaces = TypedResource[ExpandableListView](R.id.workspaces)

  val subtask_txt = TypedResource[TextView](R.id.subtask_txt)
 // ...

}
```

TR in Action

```
class MakeANoteActivity extends ScalaActivity
   with    View.OnClickListener
   with    PhotoOperations with CalibrationOperations
{
  val PhotoPath     = "picture_path"
  val PhotoAttached = "photo_attached"

  lazy val sessionManager = SessionManager.get   // magic here but
  lazy val locationManager = LocationManager.get // we'll see this later!
```

```
  lazy val attachPhoto = findView(TR.attach_photo)
  lazy val noteText    = findView(TR.note_text)
  lazy val save        = findView(TR.save_button)
  lazy val dateText    = findView(TR.date)
  lazy val share       = findView(TR.share)

  // magic here too, but we'll implement this in a few slides!
  val imAircasting = findResource[String](R.string.im_aircasting)
  val shareWith    = findResource[String](R.string.share_with)

  // ...
}
```

Types



You always have the **type information** at hand.

(**IntelliJ IDEA**, ENSIME/Emacs, Eclipse)

⌘ SHIFT I
ALT =

Menus


LOCO MOJITO
MEXICAN/TEXAN
CANTINA & COCKTAIL BAR
197 BRICK LANE, LONDON E1 6SA
020 7033 4666
Starters
1. CIABATTA GARLIC BREAD (V) £2.95
Grilled Ciabatta bread with lots of garlic butter
2. CIABATTA GARLIC BREAD WITH CHEESE & SALAD (V) £3.45
Ciabatta garlic bread topped with mild salsa
3. LOUISIANA BUFFALO WINGS £3.95
Oven roasted tender chicken wings smothered in spicy BBQ sauce
4. BBQ RIBS SAMPLER £3.95
Tender pork spare ribs in our 'soon to be famous' smoked honey BBQ sauce
5. MEXICAN SALSA DIPS (V) £4.25
Tortilla chips served with spicy tomato salsa, sour cream & guacamole
6. FIRE BEAN & CHILLI HUMMUS DIPS (V) £4.25
Delicious spicy re-fried bean & chilli hummus dips – served with tortilla chips
7. BEEF CHILLI SAMPLER £4.25
Bowl of our famous chilli on carne with sour cream, onions & tortilla chips
8. CALAMARI FRIED £4.95
Deep fried lightly battered calamari rings, served with lime & chilli tartare
9. CREAM CHEESE JALAPENOS (V) £4.75
Cream cheese stuffed with jalapeno peppers, breaded & deep fried golden, served with salsa dip
10. COLORADO MUSHROOMS (V) £4.50
Cheese & garlic stuffed mushrooms, breaded & deep fried golden, served with sour cream
11. LOADED POTATO SKINS
Served crispy with a selection of savory fillings
A) SMOKED BACON & CHEESE £3.95
B) CHILLI CON CARNE & SOUR CREAM £4.25
Nachos
12. CLASSIC NACHOS (V) £3.95
Tortilla chips topped with tomato salsa, melted cheese, sour cream, guacamole & jalapenos
13. SPICY BEEF NACHOS £4.95
Classic cheese nachos above with an additional topping of spicy beef, served with sour cream & guacamole
14. CHICKEN NACHOS £4.95
Tortilla chips topped with salsa, cheese, shredded chicken, guacamole & sour cream
15. CHORIZO & OLIVE NACHOS £4.95
A colourful range of toppings, salsa, cheese, spicy chorizo salami, black olives, sour cream & guacamole
Combo Fiesta to Share
Hey amigos, cant choose? Let's mix it up! Mixed starter consisting of smoked BBQ ribs, Louisiana chicken wings, Colorado mushrooms & cream cheese jalapenos, served with sour cream, salsa & tortilla chips
(serving for 2 £8.95 – serving for 4 £15.95)
Tacos
Crispy corn tacos (2) filled with lettuce, pico degallo, salsa, cheese & a additional topping from the list below. Served with guacamole & sour cream
17. CHILLI CHICKEN £4.95
18. FIRE BEAN & HUMMUS (V) £4.95
19. CHILLI BEEF £4.95
20. CHORIZO & CHEESE QUESADILLA £4.50
Grilled tortilla filled with chorizo, cheese, pico de gallo & salsa, served with tomato chilli relish
21 HONEY DUCK & GINGER TAQUITAS £5.20
Deep fried Mexican style tortilla rolls filled with honey duck, ginger, spring onion, bamboo shoots, salsa & jalapenos, served with sweet chilli dip
22. PERI PERI BBQ PRAWNS £5.50
King prawns skewered with limes & chargrilled with Cajun spices, served with peri peri chilli dip. TOTALLY TROPICAL!
23. MUSSELS IN TOMATO CHILLI SALSA £5.50
Bowl of jumbo mussels in a smoked chilli & garlic tomato salsa, served with sour cream & a slice of garlic bread
Drinks
FRUIT JUICE.SOFT DRINKS £2.25
MINERAL WATER (BTL 500ML) £2.00
MINERAL WATER (BTL 1 LTR) £3.50
CAPPUCINO £1.95
Note
- As fresh chickens are used in our dishes, small pieces of bone may rarely be found in your meal.
- Minimum food charge £6.00 per person on weekends & bank holidays

Menus = callbacks

Java

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
  MenuInflater inflater = getMenuInflater();
  inflater.inflate(R.menu.game_menu, menu);
  return true;
}
```

Menus = callbacks

Menus = callbacks

Java

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
  MenuInflater inflater = getMenuInflater();
  inflater.inflate(R.menu.game_menu, menu);
  return true;
}
```

```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    switch (item.getItemId()) {
        case R.id.new_game:
            newGame();
            return true;
        case R.id.help:
            showHelp();
            return true;
        default:
            return super.onOptionsItemSelected(item);
    }
}
```

Menus = callbacks

Java

```
public class AnActivity extends Activity {

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
      MenuInflater inflater = getMenuInflater();
      inflater.inflate(R.menu.game_menu, menu);
      return true;
    }


            @Override
            public boolean onOptionsItemSelected(MenuItem item) {
                switch (item.getItemId()) {
                    case R.id.new_game:
                        newGame();
                        return true;
                    case R.id.help:
                        showHelp();
                        return true;
                    default:
                        return super.onOptionsItemSelected(item);
                }
            }

}
```

```
public class AnActivity extends Activity {

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
      MenuInflater inflater = getMenuInflater();
      inflater.inflate(R.menu.game_menu, menu);
      return true;
    }


            @Override
            public boolean onOptionsItemSelected(MenuItem item) {
                switch (item.getItemId()) {
                    case R.id.new_game:
                        newGame();
                        return true;
                    case R.id.help:
                        showHelp();
                        return true;
                    default:
                        return super.onOptionsItemSelected(item);
                }
            }

}
```

Menus = callbacks

Java

```
public class AnActivity extends Activity {

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
      MenuInflater inflater = getMenuInflater();
      inflater.inflate(R.menu.game_menu, menu);
      return true;
    }


            @Override
            public boolean onOptionsItemSelected(MenuItem item) {
                switch (item.getItemId()) {
                    case R.id.new_game:
                        newGame();
                        return true;
                    case R.id.help:
                        showHelp();
                        return true;
                    default:
                        return sup
                }
            }
        }
}
```

Java-style ways to fix this?

```
public class AnActivity extends Activity {

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
      MenuInflater inflater = getMenuInflater();
      inflater.inflate(R.menu.game_menu, menu);
      return true;
    }


            @Override
            public boolean onOptionsItemSelected(MenuItem item) {
                switch (item.getItemId()) {
                    case R.id.new_game:
                        newGame();
                        return true;
                    case R.id.help:
                        showHelp();
                        return true;
                    default:
                        return super.onOptionsItemSelected(item);
                }
            }
}
```

Delegate! Delegate! Delegate!

Java

```
public class GameMenu {
  public boolean onCreateOptionsMenu(Menu menu) { /*...*/ }
  public boolean onOptionsItemSelected(MenuItem item) { /*...*/ }
}
```

```
public class AnActivity extends Activity {
  GameMenu gameMenu = ...

  public boolean onCreateOptionsMenu(Menu menu) {
    return gameMenu.onCreateOptionsMenu(menu);
  }

  public boolean onOptionsItemSelected(MenuItem item) {
    return gameMenu.onOptionsItemSelected(item)
  }


}
```

super class **monster**

Java

```
public abstract class WithGameMenuActivity extends Activity {
  public boolean onCreateOptionsMenu(Menu menu) { /*...*/ }
  public boolean onOptionsItemSelected(MenuItem item) { /*...*/ }
}
```

```
public class AnActivity extends WithGameMenuActivity {

}
```

Menus = callbacks

Java

```
public class AnActivity extends Activity {

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
      MenuInflater inflater = getMenuInflater();
      inflater.inflate(R.menu.game_menu, menu);
      return true;
    }


            @Override
            public boolean onOptionsItemSelected(MenuItem item) {
                switch (item.getItemId()) {
                    case R.id.new_game:
                        newGame();
                        return true;
                    case R.id.help:
                        showHelp();
                        return true;
                    default:
                        return su
                }
            }
}
```

Scala-style ways to fix this?

```
public class AnActivity extends Activity {

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
      MenuInflater inflater = getMenuInflater();
      inflater.inflate(R.menu.game_menu, menu);
      return true;
    }


            @Override
            public boolean onOptionsItemSelected(MenuItem item) {
                switch (item.getItemId()) {
                    case R.id.new_game:
                        newGame();
                        return true;
                    case R.id.help:
                        showHelp();
                        return true;
                    default:
                        return super.onOptionsItemSelected(item);
                }
            }
}
```

Converting is simple

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    MenuInflater inflater = getMenuInflater();
    inflater.inflate(R.menu.game_menu, menu);
    return true;
}
```

```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    switch (item.getItemId()) {
        case R.id.new_game:
            newGame(); return true;
        case R.id.help:
            showHelp(); return true;
        default:
            return super.onOptionsItemSelected(item);
    }
}
```

The Menu Trait

Scala

```
trait GameMenu {


  override def onCreateOptionsMenu(Menu menu) = {
    val inflater = getMenuInflater()
    inflater.inflate(R.menu.game_menu, menu)
    true
  }
  override def onOptionsItemSelected(MenuItem item) =
    item.getItemId() match {
      case R.id.new_game => newGame(); true
      case R.id.help     => showHelp(); true
      case _             => super.onOptionsItemSelected(item)
    }


}
```

Uhm... but we're not an Activity here!

The Menu Trait

Scala

```
trait GameMenu {
  this: Activity =>

  override def onCreateOptionsMenu(Menu menu) = {
    val inflater = getMenuInflater()
    inflater.inflate(R.menu.game_menu, menu)
    true
  }

  override def onOptionsItemSelected(MenuItem item) =
    item.getItemId() match {
      case R.id.new_game => newGame(); true
      case R.id.help     => showHelp(); true
      case _             => super.onOptionsItemSelected(item)
    }


}
```

Now we're sure **"I'm in an Activity"**

The Menu Traits

```
class MyGame extends Activity with GameMenu
                              with SomeContextMenu
```

An **Implicit** Context

Passing around **Context**

```
Toast.makeText(ctx, msg, LENGTH_LONG).show();
```

```
Intent intent = new Intent(context, LocalService.class);
```

An **Implicit** Context

```
def toastMyName(name: String, ctx: Context) {
  import android.widget.Toast._
  makeText(ctx, name, LENGTH_LONG).show()
}

toastMyName(”Siegfried”, getContext)
```

Implicit != Explicit

An **Implicit** Context

Implicit != Explicit

```
def toastMyName(name: String, ctx: Context) {
  import android.widget.Toast._

  makeText(ctx, msg, LENGTH_LONG).show()
}

toastMyName(”Siegfried”, getContext)
```

Explicit Parameter

An **Implicit** Context

Implicit != Explicit

```
def toastMyName(name: String, ctx: Context) {
  import android.widget.Toast._

  makeText(ctx, msg, LENGTH_LONG).show()
}


toastMyName(”Siegfried”, getContext)
```

Explicit Parameter

An **Implicit** Context

Implicit != Explicit

Import **inside a method**!

```
def toastMyName(name: String, ctx: Context) {
  import android.widget.Toast._

  makeText(ctx, msg, LENGTH_LONG).show()
}


toastMyName(”Siegfried”, getContext)
```

An **Implicit** Context

Implicit != Explicit

```
def toastMyName(name: String)(implicit ctx: Context){{
  import android.widget.Toast._

  makeText(ctx, name, LENGTH_LONG).show()
}       }
```

```
toastMyName("Siegfried", getContext)
```

Implicit Parameters

Implicit != Explicit

```
def toastMyName(name: String)(implicit ctx: Context) {
  import android.widget.Toast._

  makeText(ctx, msg, LENGTH_LONG).show()
}
```

Implicit Parameter

Implicit Parameters

Implicit != Explicit

```
def toastMyName(name: String)(implicit ctx: Context) {
  import android.widget.Toast._

  makeText(ctx, msg, LENGTH_LONG).show()
}

implicit val context: Context = getContext


toastMyName("Siegfried")

toastMyName("Siegfried")(context)
```

An **Implicit** Context

```
class ScalaActivity
  extends Activity
     with ImplicitContext
```

"I can be only **mixed into** an **Activity**."

```
trait ImplicitContext {
  this: Activity =>

  implicit val ctx = getContext
}
```

So I know this method!

Implicit Conversions

Implicit **Conversions**

Implicit != Explicit

Implicit **Conversions**

Implicit != Explicit

Implicit Context, remember?

```
toastMyName("Siegfried")
```

What if we could...

```
"Siegfried".toast()
```

Implicit **Conversions**

Implicit != Explicit

Decorator

Constructor

```
class Toastable(msg: String) {
  def toast()(implicit ctx: Context) = ???
}
```

```
new Toastable("Hello!").toast()
```

Implicit **Conversions**

Implicit != Explicit

```
class Toastable(msg: String) {
  def toast()(implicit ctx: Context) = ???
}
```

Implicit Conversion: String => Toastable

```
trait ScalaToasts {

  implicit def asToastable(str: String) =
    new Toastable(str)
}
```

```
asToastable("Hello!").toast()
```

Implicit **Conversions**

Implicit != Explicit

```
class Toastable(msg: String) {
  def toast()(implicit ctx: Context) = ???
}
```

Implicit Conversion: String => Toastable

```
trait ScalaToasts {

  implicit def asToastable(str: String) =
    new Toastable(str)
}
```

"Hello!".toast()

```
class ExampleActivity
  extends Activity
  with ImplicitContext
  with ScalaToasts
```

Implicit **Conversions**

Implicit != Explicit

```
class Toastable(msg: String) {
  def toast()(implicit ctx: Context) = ???
}
```

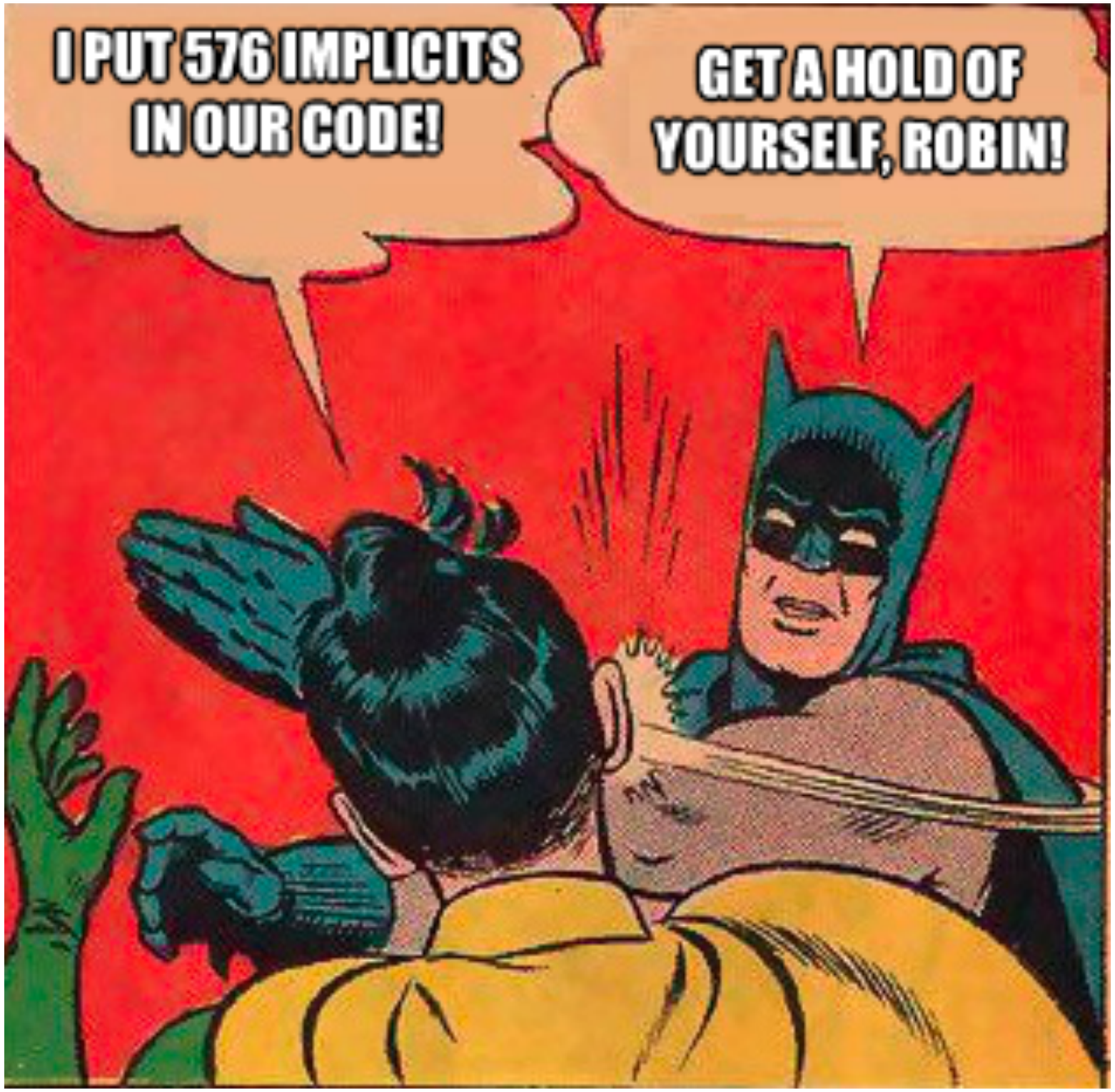
```
trait ScalaToasts {

  implicit def asToastable(str: String) =
    new Toastable(str)
}
```

```
class ExampleActivity
  extends Activity
  with ImplicitContext
  with ScalaToasts
```

"Hello!".toast()

asToastable("Hello!").toast()

Implicit **Conversions**



And there's more!

Simple Threading

```
implicit val handler = new Handler
```

```
inUiThread {
  // ...
}
```

```
inFutureWithProgressDialog(timeout = 10.seconds) {
  // ...
}
```

Implicit Conversion on **Int**

Not a replacement for proper design

on___

```
button.setOnClickListener(new View.OnClickListener() {
   @Override
   public void onClick(View view) {
      // ...
   }
});
```

```
button onClick { view =>
    // ...
}
```

SharedPreferences

```
KanbaneryPreferences.projectId = 1337
```

```
val name: Option[Long] = KanbaneryPreferences.projectId
```

SharedPreferences

```
object KanbaneryPreferences {

 private val KeyLogin  = "login"

  private def sharedPreferences(implicit ctx: Context) = ???

  def login(implicit ctx: Context) =
    sharedPreferences.getString(KeyLogin, "")
```

SharedPreferences

```
object KanbaneryPreferences {

 private val KeyLogin  = "login"

  private def sharedPreferences(implicit ctx: Context) = ???

  def login_=(number: String)(implicit ctx: Context) {
    withSharedPreferencesEditor {
      _.putString(KeyLogin, number)
    }

 }
```

SharedPreferences

```
object KanbaneryPreferences {

  private val KeyLogin  = "login"

  private def sharedPreferences(implicit ctx: Context) = ???

  def login_=(number: String)(implicit ctx: Context) {
    withSharedPreferencesEditor {
      _.putString(KeyLogin, number)
    }

}
```

SharedPreferences

```
def withSharedPreferencesEditor
    (block: SharedPreferences.Editor => Unit)
    (implicit ctx: Context) {

  val editor = sharedPreferences.edit()

  block(editor)

  editor.commit()
}
```

```
withSharedPreferencesEditor {
  _.putString(KeyLogin, number)
}
```

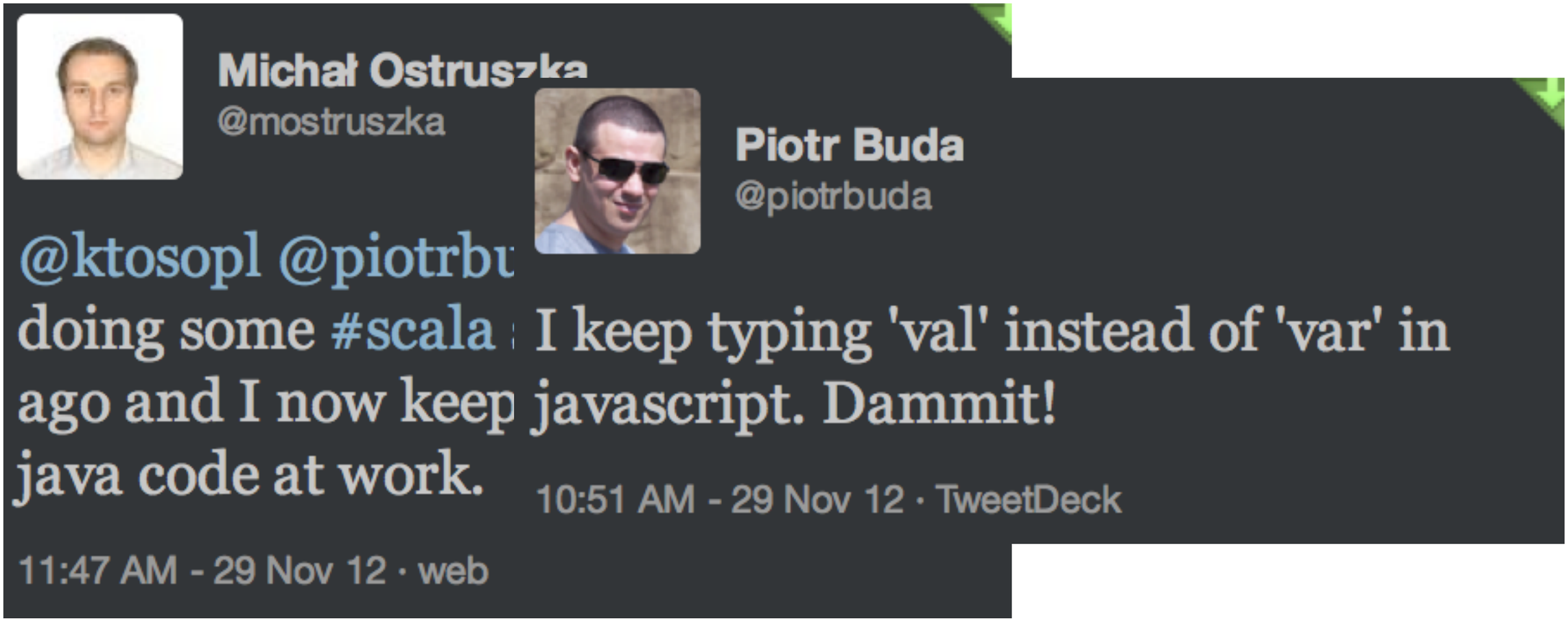
Let me warn you...

Scala is addictive.

Let me warn you...

Scala is addictive.

Scala is addictive.



Scala my Android?

Scala my Android?

Scala my Android?

- Android Library projects?
- Not "Java level" IDE support?
- Debugging?
- ProGuard-ing time?
- Increased cognitive load?

Scala my Android?

+ **No boilerplate**
+ Focus on the app/domain
+ No need for reflection
+ Less cognitive load?
+ Scala libraries
+ SBT (build tool)

def links =

- **Scala Lang** http://www.scala-lang.org/
- **Scala Koans** http://www.scalakoans.org
- **Blog.Project13.pl** - http://www.blog.project13.pl
- **SBT** Android Plugin - https://github.com/jberkel/android-plugin
- **Kanbanery for Android** - https://github.com/ktoso/kanbanery-tv
  - Check pl.project13**.scala.android.***
  - Fully Open Source
  - New version soon! **Pull** and play with it!

- **Scaloid** https://github.com/pocorall/scaloid
  - Many of the things we've seen, a bit more; some awesome, some less

Konrad Malawski / @ktosopl
GeeCON / GDG / PJUG / KSUG / SCKRK
JFokus **06.02.2013**