Mirah (née Duby)

Compilable Rubyish for JVM

Matthew Bennett
eShopworks Ltd.
@undecisive
matthew@quickwebdesign.net


OR...

Mirah + Pindah

Making Android development not suck

And look kinda rubyish and kinda nice and maybe

actually approachable to people who give a damn about beautiful readable and expressive code without all the overhead of JRuby and standard libraries

Matthew Bennett
eShopworks Ltd.
@undecisive
matthew@quickwebdesign.net

JVM... Without Java!

... (but no standard library either)

Matthew Bennett - matthew@quickwebdesign.net

Why Mirah?

- JVM
  - Portable
  - Fast
  - NOT Java!
- Android made better
  - No Java
- Alternatives?
  - Ruboto / jRuby: too much overhead
  - Ruboto-core: Another promising possibility
  - Scala, Clojure, Groovy, etc. Learning curve

Originally written by Charles Nutter of jruby fame...

What you will need

- Java, Ant & JRuby
- Javaphobia counselling
  - API
- A little patience
  - It's only alpha

"And how does that make you feel?"

- This presentation was tested in Mirah v0.0.7, Pindah v0.1.0

Installation

$ jruby -S gem install mirah

- Note: Don't use rvm gem
- Current is 0.0.7


Oh, and did I mention...
strong type checking.
(bleuch)

Some Ruby Code

```
class Rubyish
  def a_method(some_string, some_hash)
    @message = some_string
    @transform = some_hash
    puts @message
  end

  def another_method
    @transform.each_pair do |key, val|
      puts @message.gsub(key.to_s, val.to_s)
    end
  end
end


@a = Rubyish.new
@a.a_method("This looks like ruby", :looks => :smells)
@a.another_method
```

Some Mirah Code

```
import java.util.HashMap

class Rubyish
  def a_method(some_string:String, some_hash:HashMap)
    @message = some_string
    @transform = some_hash
    puts @message
  end

  def another_method
    @transform.keys.each do |key|
      puts @message.replaceAll(String(key),
                               String(@transform[key]))
    end
  end
end



@a = Rubyish.new
@a.a_method("This looks like ruby", :looks => :smells)
@a.another_method
```

Matthew Bennett - matthew@quickwebdesign.net

Some Mirah Code

```
import java.util.HashMap

class Rubyish
  def a_method(some_string:String, some_hash:HashMap)
    @message = some_string
    @transform = some_hash
    puts @message
  end

  def another_method
    @transform.keys.each do |key|
      puts @message.replaceAll(String(key),
                               String(@transform[key]))
    end
  end
end



@a = Rubyish.new
@a.a_method("This looks like ruby", :looks => :smells)
@a.another_method
```

Statically Typed

UUUUGGGGGLLLLY

Matthew Bennett - matthew@quickwebdesign.net

Some Mirah Code

```
import java.util.HashMap

class Rubyish
  def a_method(some_string:String, some_hash:HashMap)
    @message = some_string
    @transform = some_hash
    puts @message
  end

  def another_method
    @transform.keys.each do |key|
      puts @message.replaceAll(key.toString,
                               @transform[key].toString)
    end
  end
end

@a = Rubyish.new
@a.a_method("This looks like ruby", :looks => :smells)
@a.another_method
```

UUUUGGGGGLLLLY (a.k.a. “Java”)

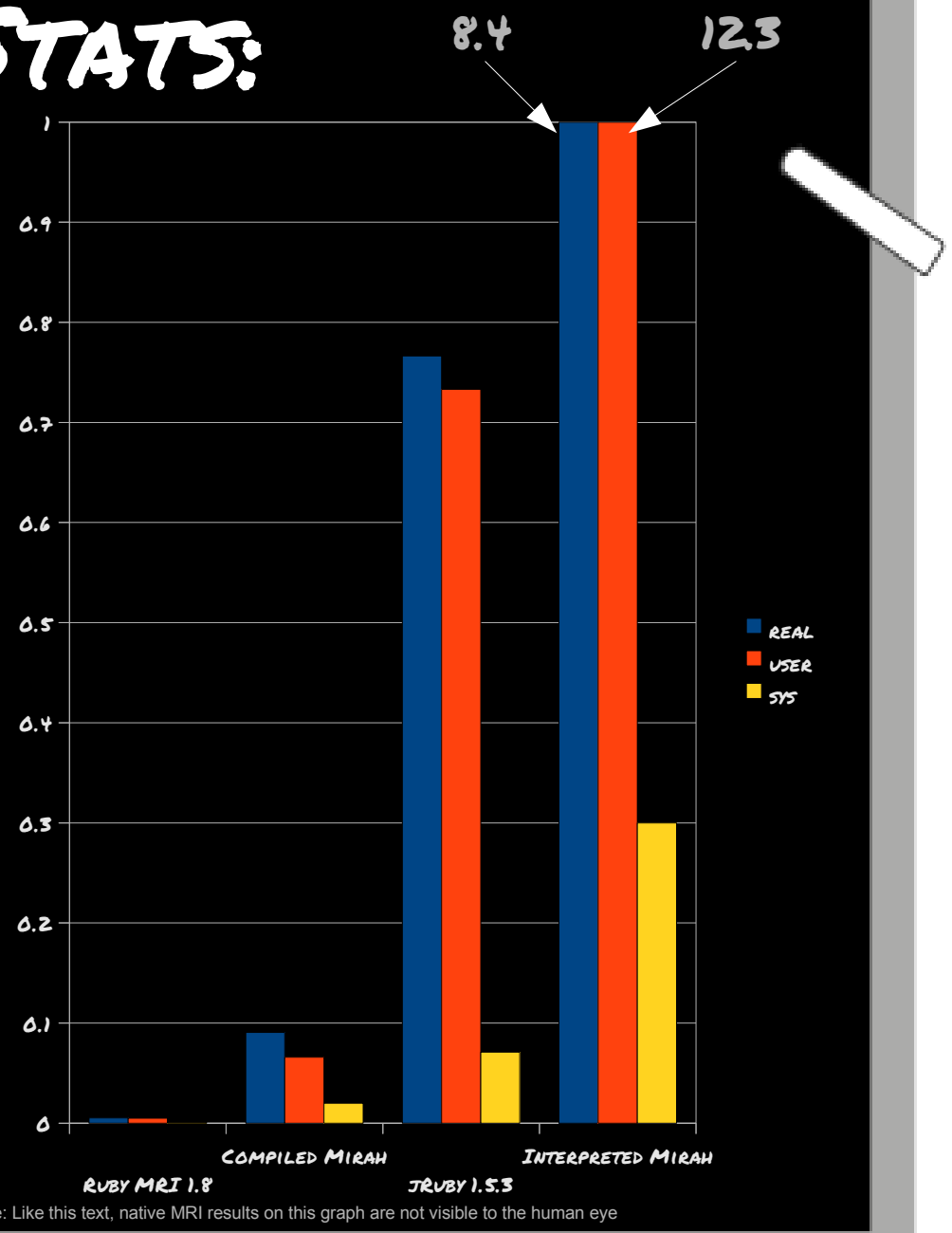
Matthew Bennett - matthew@quickwebdesign.net

Some Stats:

- Mirah vs Jruby
- For small pieces of code like the one above, running Mirah as a scripting language is silly
- Output:

```
This looks like ruby

This smells like ruby
```



Note: Like this text, native MRI results on this graph are not visible to the human eye

Matthew Bennett - matthew@quickwebdesign.net

BTW...

- This ain't no ruby...
  - Objects behave like java (e.g. closed)
  - Ruby object model fudges (def self.a)
- No return types?
  - e.g. “public static void a_method()”
  - Mirah is ~~good at~~ guessing
  - You often have to give her a helping nudge
- Have a play yourself
  - https://gist.github.com/863152

(as good as it can be while remaining lightweight...)

By helping nudge...

- def my_method(argument:String):String
- def my_method(arg:string)

  returns String
- First syntax is preferred

A few warnings for the squeamish:

- Bad error / warning messages
  - Inference error all the way
    - “Unable to infer type”. A lot. Especially on instance vars
- You will need the Java API handy
- Everyday niceties don't exist
  - e.g. arr[0] – forget it! (hash[:key] works though)
  - arr.get(0) works instead

Still in flux

- While writing this presentation (0.0.5-7)...
  - “public static void main” in compiled classes
  - Odd issues when importing java classes
- Some improvements
  - Implementing Java interfaces works!
  - If some of the problems in this presentation don't occur, they've probably been fixed!

Blocks

- You can do blocks... but you can't pass parameters

```
my_method { code } # Works*
my_method { | n | code } # No chance&
def my_method(blk:Runnable) ... blk.run
```

- *Scoping is a little twisted
  - Local vars, no instance vars
- &Array.each has been implemented (see previous example)
  - They cheated and used macros

Splatted + Optional Args

- Not possible
- Splats: *def f(a, *b)...*

```
def self.blah(args:java.util.List)
    args.each { |food| puts food}
end
blah(["fish", "chips"])
```

- Optional args: *def f(a, b='nope')*
  - Use inheritance or method overloading

And a few other things...

- Java public instance vars
  - Solution: write an accessor in Java?
- Mirah private methods (Just android / compiled?)
- Ranges (e.g. 1..100)
- No standard (rubyish) library
  - Some of the built-in macros help
  - I'd love to make one though...


Macros
(it's complicated)

But what if I like Java?

- (grumble... muttter.... well, I suppose...)
- You can compile down to Java!
  - mirahc -j file.mirah
  - Don't expect it to be pretty!
- You can compile down to specific JVMs
  - mirahc --jvm 1.4

How unpretty?

Aren't there laws against this kind of thing?

```
// Generated from rubyishc.rb
public class Rubyishc extends java.lang.Object {
  private java.lang.String message;
  private java.util.HashMap transform;
  public java.io.PrintStream a_method(java.lang.String
some_string, java.util.HashMap some_hash) {
    this.message = some_string;
    this.transform = some_hash;
    java.io.PrintStream temp$1 = java.lang.System.out;
    temp$1.println(this.message);
    return temp$1;
  }
  public java.lang.Object another_method() {
    java.util.Iterator __xform_tmp_1 = null;
    java.lang.Object key = null;
    java.util.Iterator temp$3 = null;
    java.util.Set temp$4 = null;
    {
      java.util.HashMap self$2052 = this.transform;
      temp$4 = self$2052.keySet();
    }
    temp$3 = temp$4.iterator();
    __xform_tmp_1 = temp$3;
    label1:
    while (__xform_tmp_1.hasNext()) {
      key = __xform_tmp_1.next();
      label2:
      {
        java.io.PrintStream temp$5 = java.lang.System.out;
        java.lang.String temp$6 = null;
        java.lang.String temp$7 = ((java.lang.String)(key));
        java.lang.String temp$8 = null;
        java.lang.Object temp$9 = null;
        {
          java.util.HashMap self$2054 = this.transform;
          java.lang.Object temp$10 = null;
          {
            temp$10 = key;
          }
          temp$9 = self$2054.get(temp$10);
        }
        temp$8 = ((java.lang.String)(temp$9));
        temp$6 = this.message.replaceAll(temp$7, temp$8);
        temp$5.println(temp$6);
      }
    }
    return null;
  }
```

Matthew Bennett - matthew@quickwebdesign.net

How unpretty?

```
public static void main(java.lang.String[] args) {
  Rubyishc a = null;
  a = new Rubyishc();
  java.util.HashMap temp$1 = null;
   {
     {
      java.util.HashMap self$2056 = new java.util.HashMap(16);
      java.lang.String temp$2 = null;
       {
        temp$2 = "looks";
       }
      java.lang.String temp$3 = null;
       {
        temp$3 = "smells";
       }
      self$2056.put(temp$2, temp$3);
      temp$1 = self$2056;
     }
   }
   a.a_method("This looks like ruby", temp$1);
   a.another_method();
  }
}
```

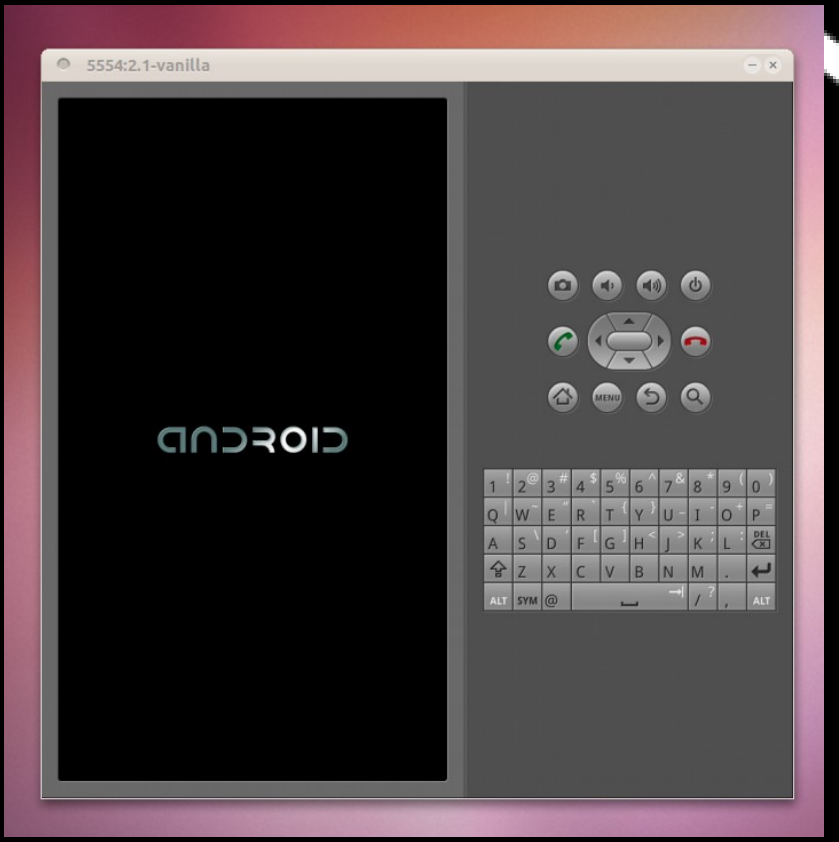
Java produced by mirah version 0.0.5 did not compile

Android Development

With Pindah

Matthew Bennett - matthew@quickwebdesign.net

For this section, you will need:



- Android SDK

- Shell setup

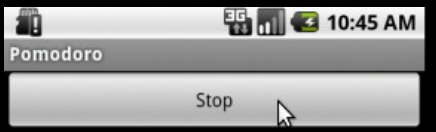
```
export JAVA_HOME=/usr/lib/jvm/java-6-sun-1.6.0.24/
export CLASSPATH=${JAVA_HOME}/lib/tools.jar
```

Installation

- Install it just like you installed mirah...
  - jruby -S gem install pindah
- Create a new project...
  - pindah create net.quickwebdesign.pomodoro

Example app

- “Pomodoro”
- Initially XML-free...
  - Failed!
  - Too much complexity for an android n00b
  - layout_width, layout_height, LayoutParams...



02:27

Starting...

Pindah

- Simply provides the basics:
  - Phil Hagelberg (technomancy)
  - Rake tasks → Ant
    - e.g. rake install / debug / clean etc
    - adb is a bit sleepy
  - Uses Android SDK for the lot
    - You know what that means?


<XML>

How bad could it be?

- res/layout/main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
  android:orientation="vertical"
  android:layout_width="fill_parent"
  android:layout_height="fill_parent">
  <Button android:id="@+id/start"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="Start Timer" />
  <LinearLayout android:id="@+id/centerer"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:gravity="center_horizontal|center_vertical">
    <Chronometer android:id="@+id/chronometer"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content" />
  </LinearLayout>
</LinearLayout>
```

In Depth: Layouts

```
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">
        ...
</LinearLayout>
```

In Depth: Nested Layouts

```
<LinearLayout android:id="@+id/centerer"
  android:layout_width="fill_parent"
  android:layout_height="fill_parent"
  android:gravity="center_horizontal|center_vertical">
        ...
</LinearLayout>
```

In Depth: Button + Chronometer

```
<Button android:id="@+id/start"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="Start Timer" />

<Chronometer
android:id="@+id/chronometer"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" />
```

You don't wanna do it like that!

- Droid-views
  - https://github.com/objo/droid-views
  - Joe O'Brien
  - Requires nokogiri
    - jruby -S gem install nokogiri

My Droid view...

```
linear_layout do |ll|
  ll.orientation    = :vertical
  ll.layout_width  = :fill_parent
  ll.layout_height = :fill_parent

  button do |butt|
    butt.id = '@+id/start'
    butt.layout_width  = :fill_parent
    butt.layout_height = :wrap_content
    butt.text = "Start Timer"
  end
```

```
linear_layout do |nest|
  nest.id = '@+id/centerer'
  nest.orientation    = :vertical
  nest.layout_width  = :fill_parent
  nest.layout_height = :fill_parent
  nest.gravity = 'center_horizontal|
                  center_vertical'

  chronometer do |ch|
    ch.id = '@+id/chronometer'
    ch.layout_width  = :fill_parent
    ch.layout_height = :wrap_content
  end
end
```


Life is not all Linear
Don't overuse it

On to the interesting bit...

- Basic android app in mirah

```
class Pomodoro < Activity
  def onCreate(state)
    super(state)
    setupToaster
    setContentView R.layout.main
    setupListeners
  end
...
end
```

Écouté s'il vous plait...

```
def setupListeners
  button = Button(findViewById R.id.start)
  button.setOnClickListener Clicker.new(self)
end

def setupToaster
  Toaster.context = self
end
```

Click me! Pleeeeeese?

```
class Clicker
  implements OnClickListener
  def initialize(activity:Activity)
    @context = activity
  end

  def onClick(v:View):void
    button = TextView(v)
    chronos = Chronometer(
                  @context.findViewById R.id.chronometer)
    if button.getText == "Stop"
      chronos.stop
      button.setText "Start"
    else
      chronos.start
      button.setText "Stop"
    end
  end
end
```

Mmmm.... Toasty

- Not strictly necessary, but fun...

```
class Toaster
  def self.context
    @context
  end

  def self.context=(other:Activity)
    @context = other
  end

  def self.toast(msg:Object)
    Toast.makeText(@context.getApplicationContext,
msg.toString, Toast.LENGTH_LONG).show();
  end
end
```

I'm a very very naughty boy...

- All that was in one file
- That file is 62 lines short
  - (Including Toaster. No kitchen sink provided.)
- It doesn't really work very well...

Pitfalls

- Some things don't work right...
- ...ADB is your friend
  - adb logcat

```
I/ActivityManager(   54): Starting activity: Intent { act=android.intent.action.MAIN
cat=[android.intent.category.LAUNCHER] flg=0x10200000
cmp=net.quickwebdesign.pomodoro/.Pomodoro bnds=[125,446][235,564] }
I/ActivityManager(   54): Start proc net.quickwebdesign.pomodoro for activity
net.quickwebdesign.pomodoro/.Pomodoro: pid=370 uid=10030 gids={1015}
I/ActivityManager(   54): Displayed activity
net.quickwebdesign.pomodoro/.Pomodoro: 1513 ms (total 1513 ms)
D/dalvikvm(  101): GC freed 2448 objects / 141304 bytes in 117ms
```

References

- I thoroughly recommend you bookmark:

  http://www.mirah.org/

  https://github.com/mirah/mirah

  https://github.com/mirah/pindah

  http://download.oracle.com/javase/6/docs/api/index.html

  http://developer.android.com/reference

  https://gist.github.com/863152 (My code)

Any Questions?

Comments / Criticizisms / Insults welcome

http://www.cs.purdue.edu/homes/dec/essay.criticize.html

All negative feedback to mailto:root@localhost

Matthew Bennett - matthew@quickwebdesign.net

Feedback from Q+A

- It sounds like most of the features we love about ruby aren't supported...
  - (Answer from audience) Yeah, but then half the things we hate about Java are missing too...
  - Mirah is still evolving. Watch this space.
- I'm comfortable with Java. Why use this?
  - You probably wouldn't. At least not yet.
  - Depending on your level of Javaphobia, Mirah strikes a nice balance.