Reactive Systems with Akka and Java

Twitter: @dhinojosa
Email: dhinojosa@evolutionnext.com

Who is this for?

Java/Groovy/Scala Developers interested in Akka

In One Sentence..

“To help the audience understand Actors, Supervision, Futures, Routers in Java”

A Note About My Presentation Style

Try to achieve the right mix of slides and code

http://github.com/dhinojosa/akka-study.git

Code that you can use and reference long after the presentation

About Akka...

"Swedified" from Áhkká

Set of Libraries to Create.. Concurrent, Fault Tolerant, and Scalable Applications

http://akka.io

Actors, Location Transparency, Finite State Machines, Actor Persistence, Clustering, Agents

Same VM or Different VM

Akka Supersedes Scala Actors

Making the case for Akka

No More Headaches From...

```
synchronized
wait
notifyAll
```

Failure Tolerance

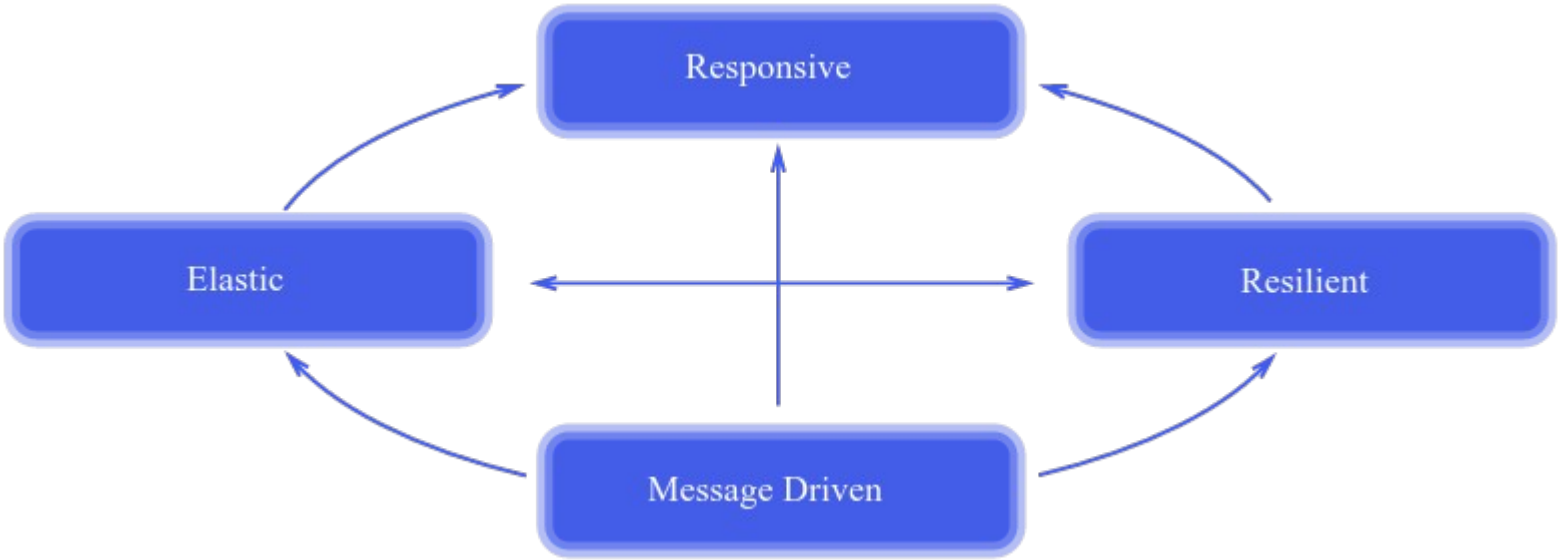
Scaling Up and Sideways

Divide and Conquer

Microservices

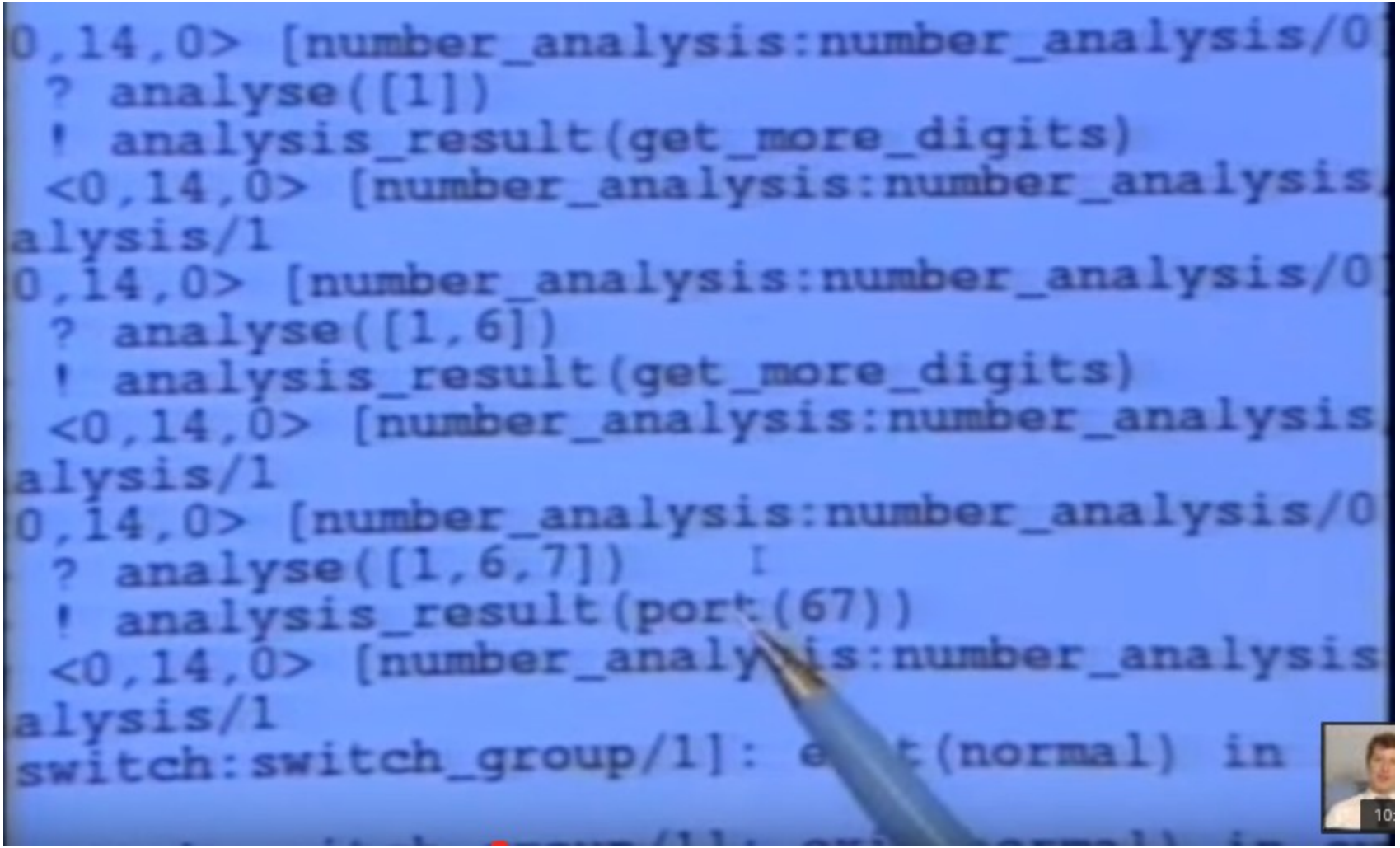
About Reactive

http://www.reactivemanifesto.org/


Responsive
Elastic
Resilient
Message Driven

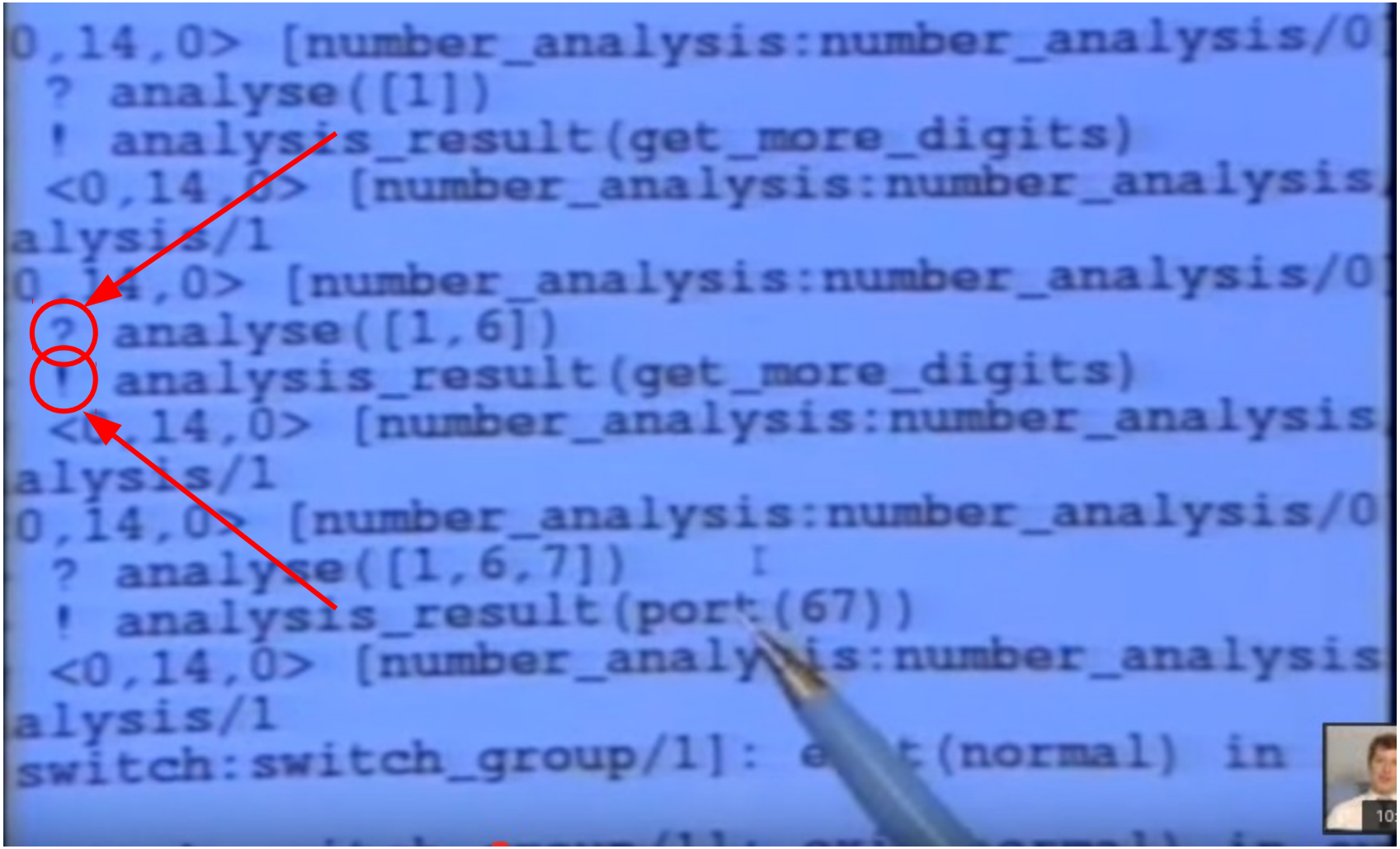
About Actors...

ERLANG



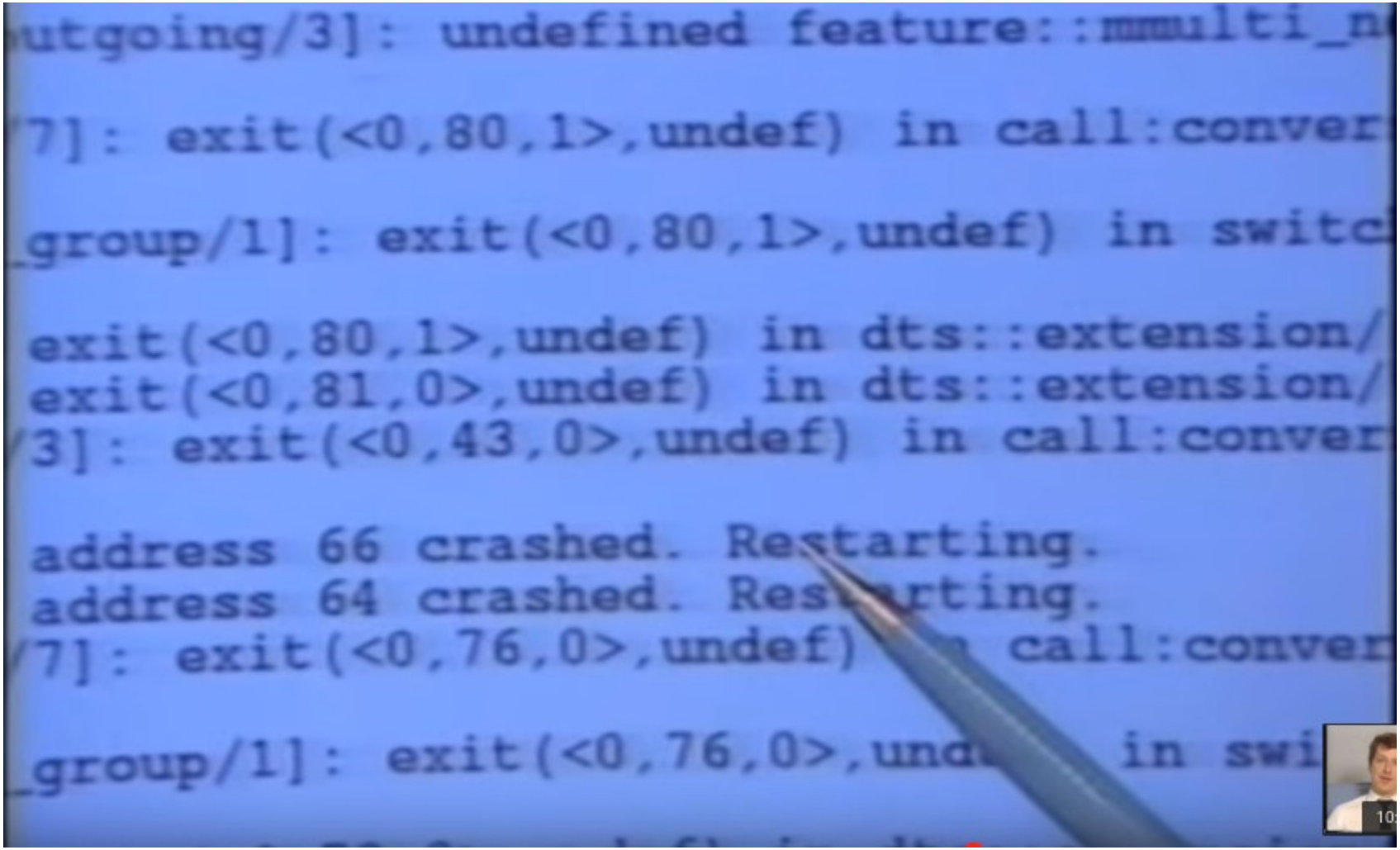
Erlang the Movie!

https://www.youtube.com/watch?v=xrIjfIjssLE



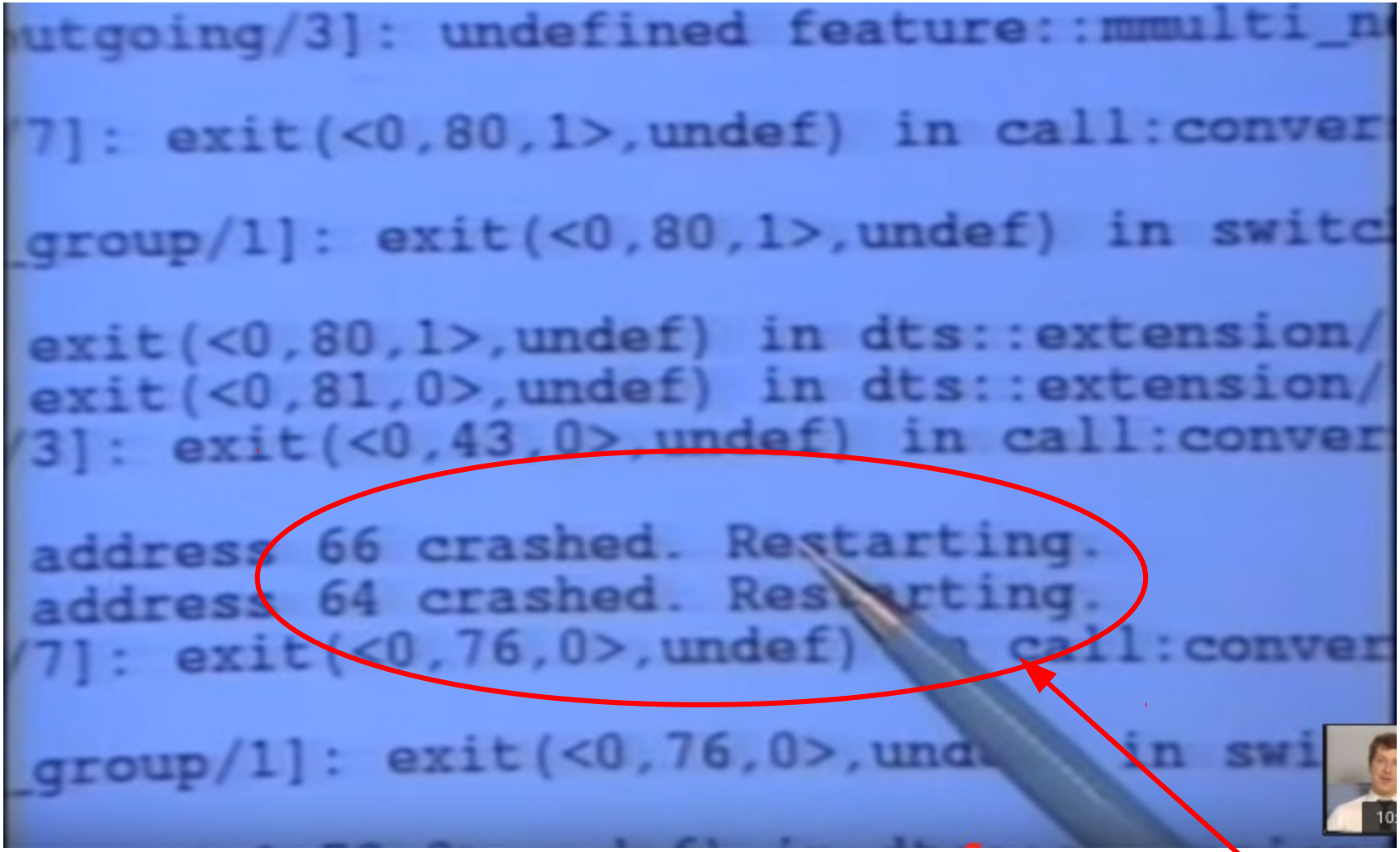
Erlang the Movie!

https://www.youtube.com/watch?v=xrIjfIjssLE



Erlang the Movie!

https://www.youtube.com/watch?v=xrIjfIjssLE



Erlang the Movie!

https://www.youtube.com/watch?v=xrIjfIjssLE

Encapsulate State and Behavior

Concurrent processors that exchange messages

Inside the Actor's Studio...

TO: MATT


TO: MATT


TO: MATT

Flooding Matt Damon with Messages

TO: MATT


TO: MATT


TO: MATT


TO: MATT


TO: MATT
TO: MATT


TO: MATT


TO: MATT


TO: MATT

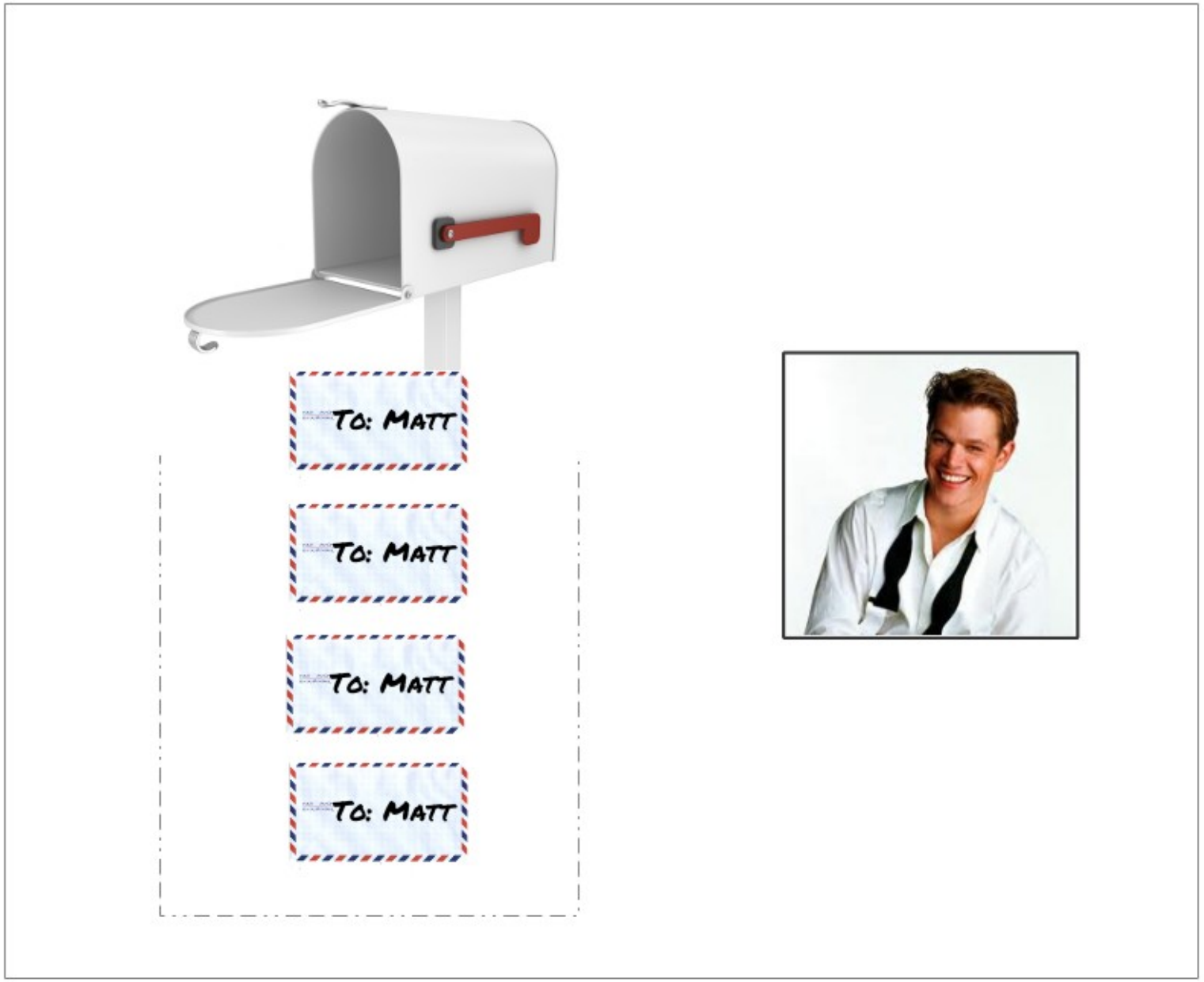

TO: MATT
TO: MATT
TO: MATT

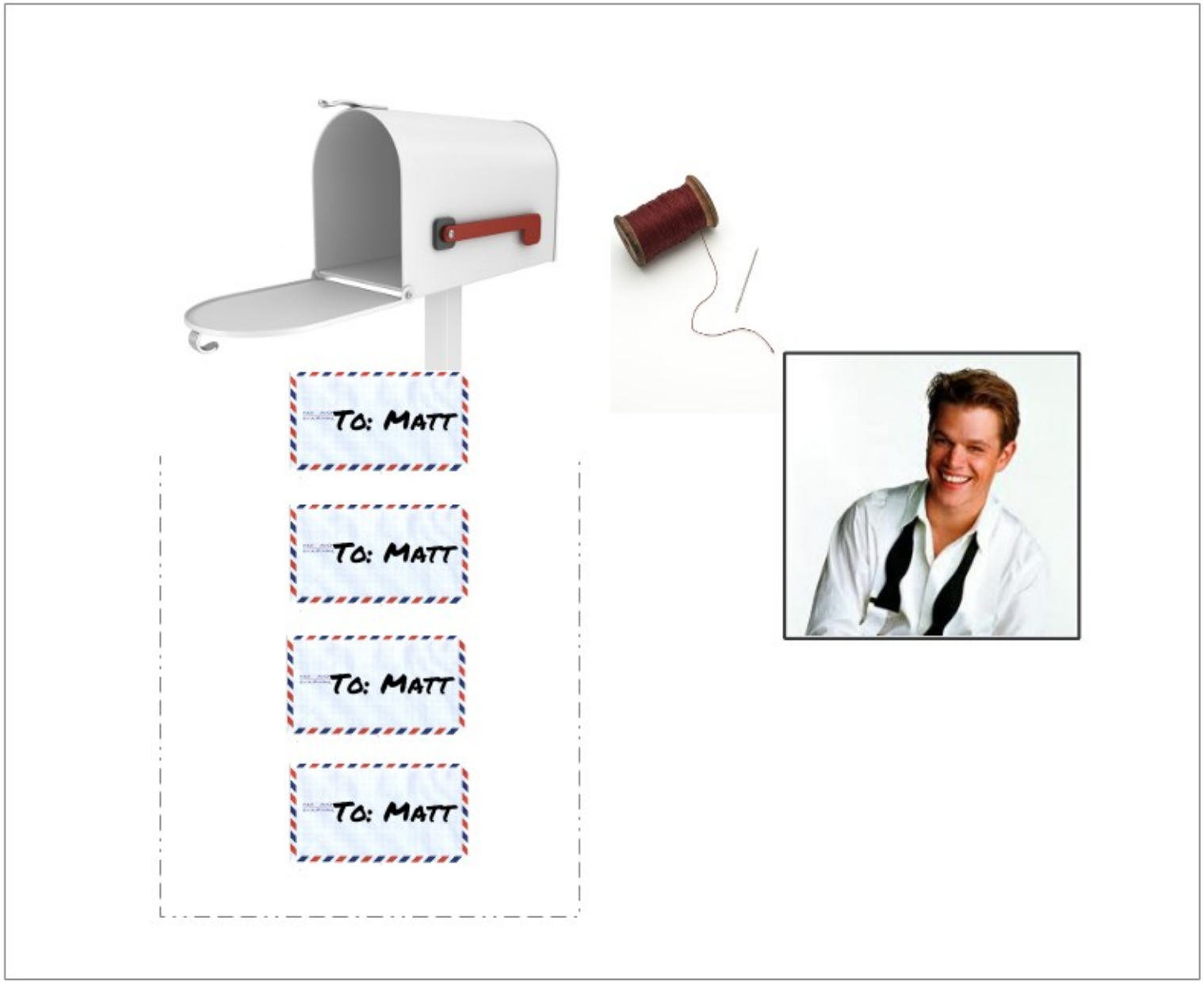

TO: MATT


TO: MATT


TO: MATT


TO: MATT


TO: MATT
TO: MATT
TO: MATT
TO: MATT


TO: MATT
TO: MATT
TO: MATT
TO: MATT


TO: MATT
TO: MATT
TO: MATT


TO: MATT
TO: MATT


TO: MATT

Message Rules

Messages must be immutable

Messages should not be a closure

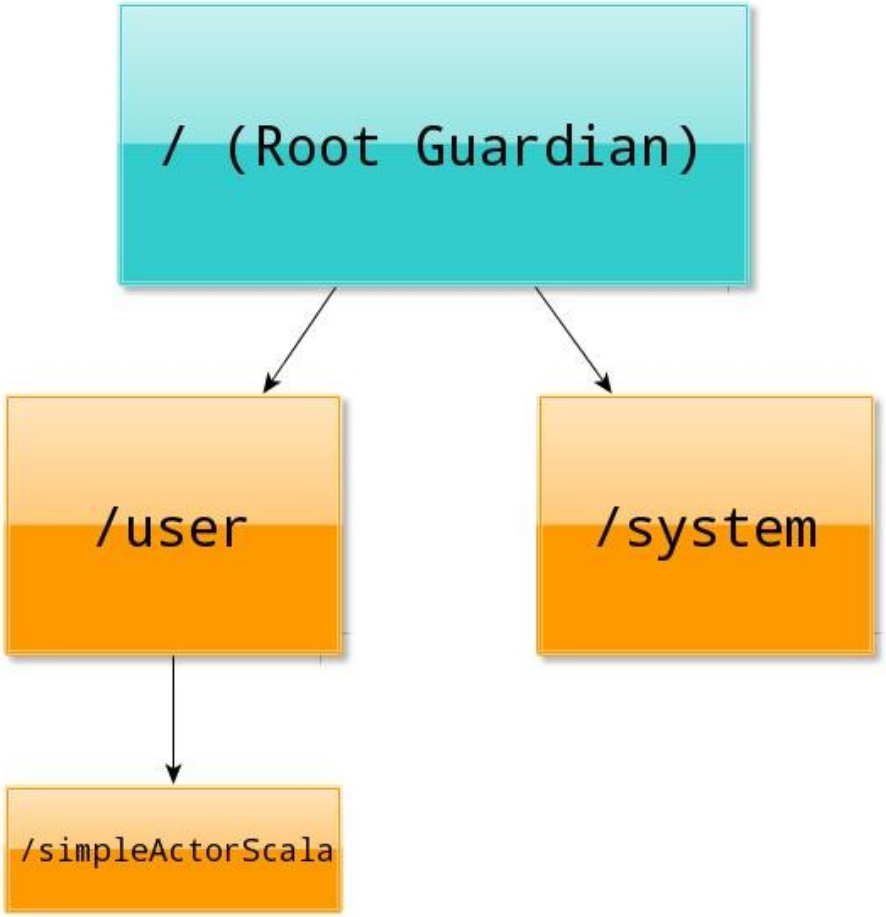
Demo: Immutable and Closures

Demo: Setting Up Actors

Review of `actorOf()`

- Creates an actor onto a `ActorSystem`
- Creates an actor given the a set of `Props` to describe the `Actor`
- Creates an `Actor` with an identifiable name, if provided
- Returns an `ActorRef`
- If `actorOf` is called inside of another actor, that new actor becomes a child of that actor

Anatomy of the Actor System



akka://{system.name}/user/{actor.name}

Paths Can Be....

Local:
`akka://my-sys/user/service-a/worker1`

Remote:
`akka://my-sys@host.example.com:5678/user/service`

Clustered:
`cluster://my-cluster/service-c`

Relative:
`../sibling-actor`

Demo: Looking Up Actors

Review of `actorSelection()`

- Was `actorFor()`.Which is now deprecated
- Actor references can be looked up using `actorSystem.actorSelection(...)` method.
- `actorSystem.actorSelection(...)` returns a `ActorSelection` object that abstracts over local or remote reference.
- `ActorSelection` can be used as long as the actor is alive.
- Only ever looks up an existing actor, i.e. does not create one.
- The actor must exist or you will receive an `EmptyLocalActorRef`
- For Remote Actor References, a search by path on the remote system will occur.

Other References

- Pure Local Actor References
- Local Actor References
- Local Actor References with Routing
- Remote Actor References
- Promise Actor References
- Dead Letter Actor References
- Cluster Actor References

Actor References within an Actor

- `self()` reference to the `ActorRef` of the actor
- `sender()` reference sender `Actor` of the last received message, typically used as described in Reply to messages
- `context()` exposes contextual information for the actor and the current message

Demo: Using Actor References inside of Actors

Demo: Futures

Demo: Ask Actors

HOCON

"Human-Optimized Config Object Notation"

HOCON

- Uses Typesafe Configuration Library
  `https://github.com/typesafehub/config`
- JSON Like Features
- Typical configuration file:
  `application.conf`

Sample HOCON Configuration

```
db.default.driver=org.h2.Driver
db.default.url="jdbc:h2:mem:play"
db.default.user=sa
db.default.password=""
```

Sample HOCON Configuration

```
db {
  default.driver=org.h2.Driver
  default.url="jdbc:h2:mem:play"
  default.user=sa
  default.password=""
}
```

Sample HOCON Configuration

```
db {
  default {
    driver=org.h2.Driver
    url="jdbc:h2:mem:play"
    user=sa
    password=""
  }
}
```

`application.conf`

- Contains Settings for Actor Systems
- All HOCON (Human Optimized Config Object Notation)
- One `application.conf` per application
- Typically stored `src/main/resources`

Demo: Review HOCON

Demo: Creating a Remote System

Fault Tolerance

Fault Tolerance

- Each actor
  - Can be a parent, and create children
  - Only an actor can create a child
  - Is responsible for their children (a.k.a. subordinate)
- When failure occurs (exceptions are thrown)
  - A child or all children are suspended
  - Parent determines the next course of action
  - Mailbox contents are maintained

Fault Tolerance Strategies

One for One Strategy

- Each child is treated is separately
- Typically the normal one that should be used
- Default if no strategy is defined

All for One Strategy

- Each child will be given the same treatment
- If one fails, they all essentially "fail"
- Used if tight coupling between children is required

Default Tolerance Behaviors

- `ActorInitializationException` will stop the failing child actor
- `ActorKilledException` will stop the failing child actor
- `Exception` will restart the failing child actor
- Other types of `Throwable` will be escalated to parent actor

Demo: Fault Tolerance

Routers

- Reroutes Messages to Other Actors
- Plenty of Prepackaged Routers
- Create Your Own Within an Existing Actor

Out of the Box Routers

- `RoundRobinRoutingLogic`
  - Round Robin Distribution
  - Chooses either number of Routees or a list of Routees not both.
- `RandomRoutingLogin`
  - Chooses a routee randomly

Out of the Box Routers

- `SmallestMailboxRouter`
  - Chooses non-suspended routee with the least messages in its mailbox
- `BroadcastRouter`
  - Sends messages to all the routees
- `ScatterGatherFirstCompletedRouter`
  - Sends messages to all routees, gets a Future,
  - Whichever routee responds first, that reponse will be sent to the sender()

Demo: Routers

Thank You

https://github.com/dhinojosa/fivelanguages

https://twitter.com/dhinojosa

http://gplus.to/dhinojosa

http://www.linkedin.com/in/dhevolutionnext