functional scala 2024
Presents

The Actor Model

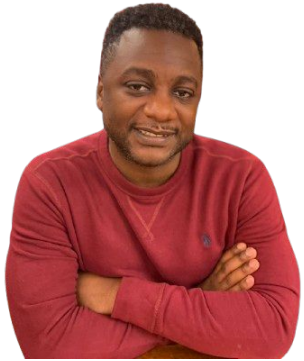
Beyond Akka With Kyo

Sponsored by

GOLEM

ZIVERGE

About Me



Distinguished Engineer

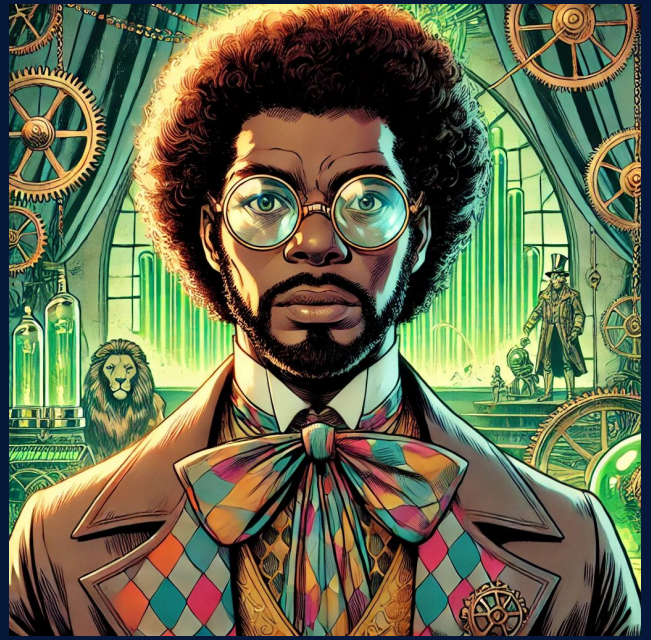
- Enthusiast & Curious Learner
  - Functional Programming
  - Programming Languages
  - Distributed Systems
- 5 time Functional Scala Speaker
- Repping: CT/NY/Texas by way of the island of Jamaica

How'd We Get Here?

So admittedly I'm an enthusiastic, curious, out the box kind of thinker...

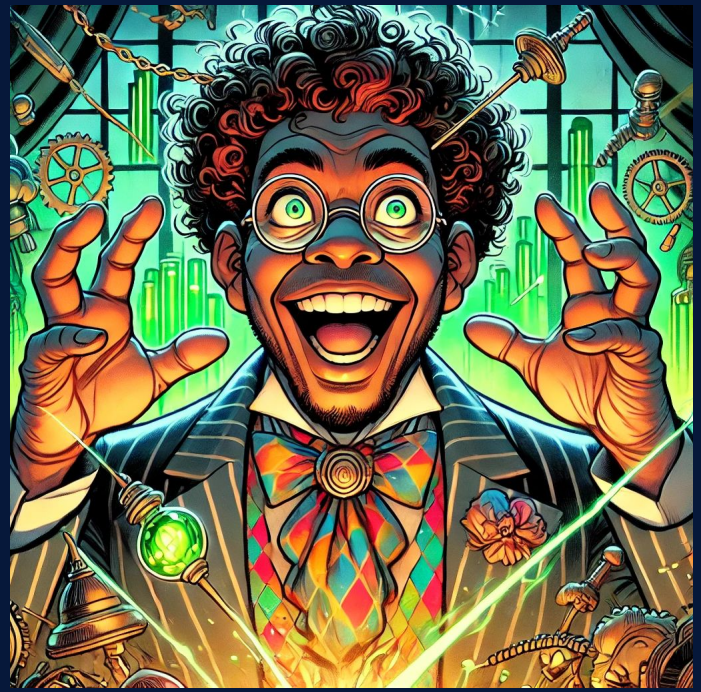
Maybe a bit of a tinkerer...

Maybe considered by some a bit of a wiz[ard]...



How'd We Get Here?

But in some cases perhaps I'm more mad scientist, than a reasoned and measured enterprise software developer and open source maintainer.



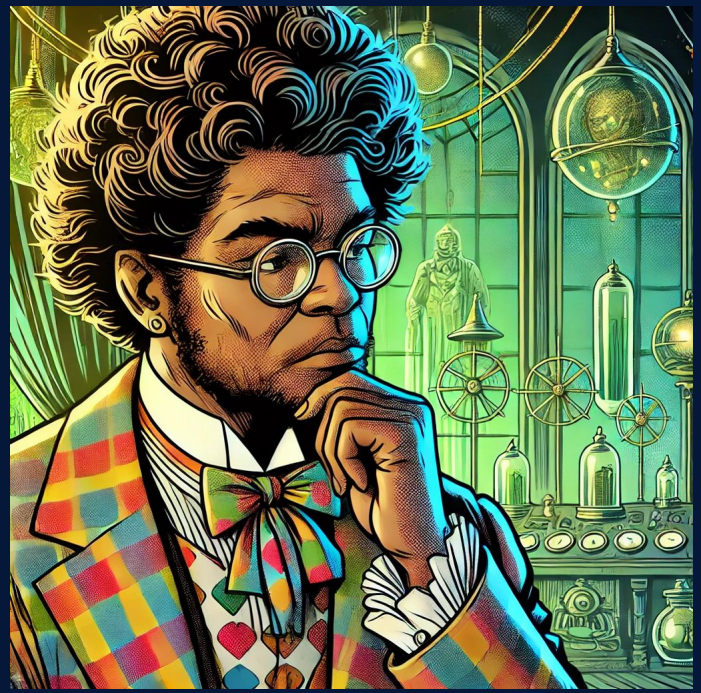
How'd We Get Here?

I had a problem.

There were tried and true, established and tested solutions.

But I could not escape the thought:

"How could I use this as an opportunity to learn and innovate?"



The Journey Ahead

01

Setting The Stage

A bit about the Actor Model

02

If I Only Had A Brain

Learning and thinking

03

Take Heart

This will make sense... don't worry

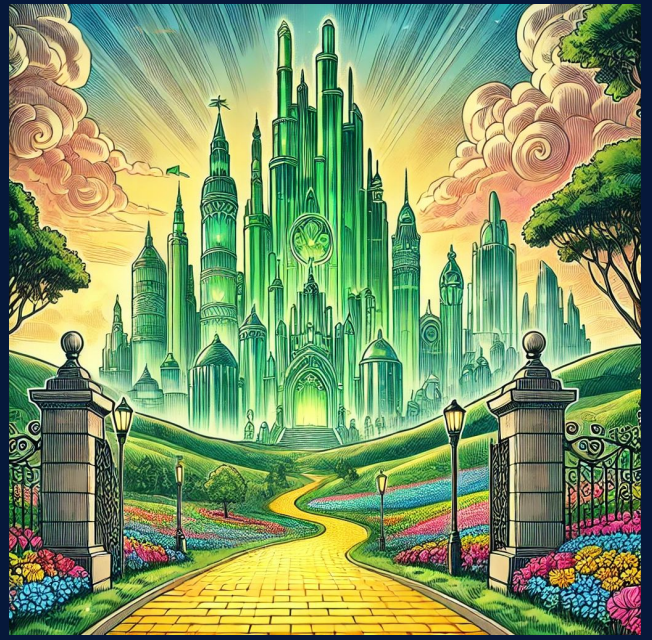
04

Be Brave

Running with scissors is fine

1. Setting The Stage

We Are Embarking On A Journey

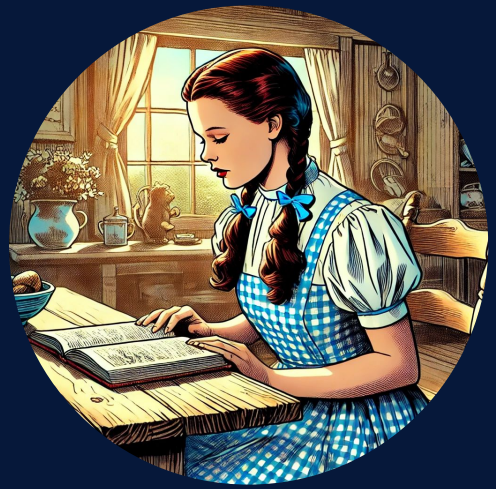


Questions?

- What Is The Actor Model?
- What Are Some Implementations?
- What Would It Take To Implement Our Own Version?

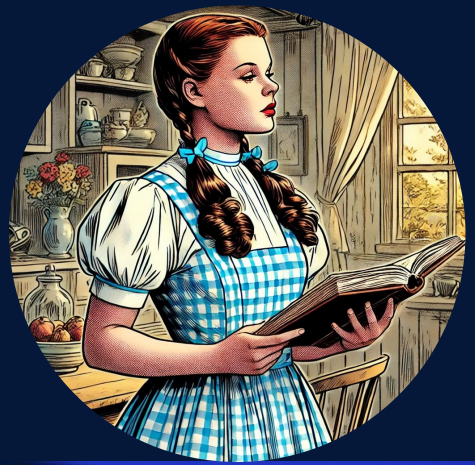
What is the Actor Model?

The Actor Model is a conceptual model for handling concurrent computation. It was introduced in 1973 by Carl Hewitt, along with Peter Bishop and Richard Steiger, as a way to think about distributed systems and parallel processing.



What is the Actor Model?

- Concurrency Model
- No Shared State
- Message Passing
- Non-blocking and Asynchronous



What is the Actor Model?

Concurrency Model

Send Messages

Actors can send messages to other Actors

Create New Actors

Actors can create new Actors

Change Behavior

An actor can change its own behavior based on the messages it receives.

What is the Actor Model?

No Shared State

One of the main principles of the Actor Model is that actors do not share state. Each actor has its own private state, and communication happens exclusively through message passing. This makes the Actor Model inherently thread-safe and well-suited for handling concurrency and distributed systems.

What is the Actor Model?

Message Passing

In the Actor Model, actors communicate by passing messages asynchronously. When an actor receives a message, it can take actions, including creating more actors, sending messages to other actors, or updating its internal state.

What is the Actor Model?

Non-Blocking & Asynchronous

Actors execute independently of each other, and communication is non-blocking. This asynchronous, decoupled nature makes it particularly suitable for scalable and distributed computing.

Prior Art

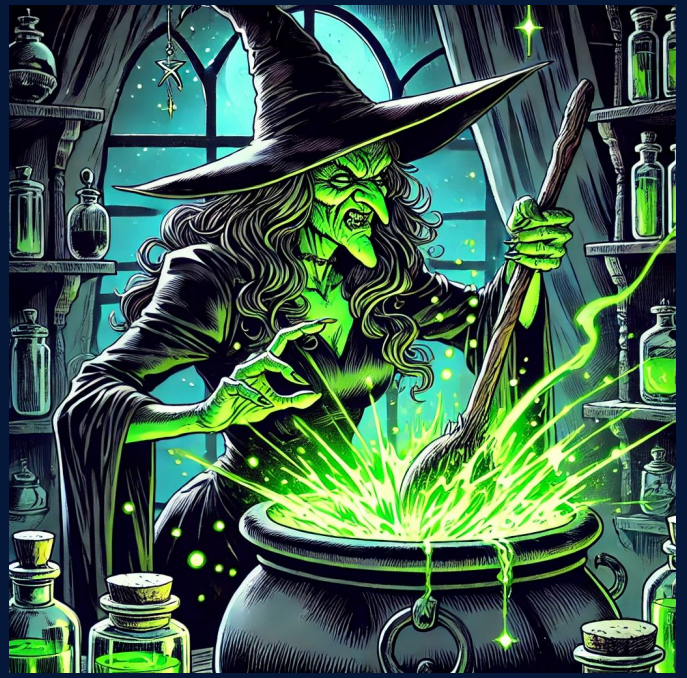
akka

Apache Pekko™

Orleans

elixir

com.lihaoyi/castor

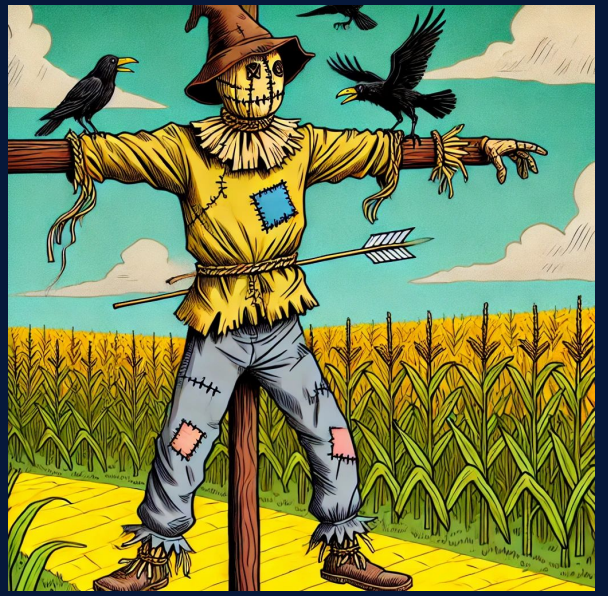


2. If I Only Had A Brain

Learning and Thinking

Questions?

- What Is Kyo?
- What Primitives Does It Offer?



What is the Kyo?

Kyo is a toolkit for Scala development, spanning from browser-based apps in ScalaJS to high-performance backends on the JVM. It introduces a novel approach based on algebraic effects to deliver straightforward APIs in the pure Functional Programming paradigm. Unlike similar solutions, Kyo achieves this without inundating developers with concepts from Category Theory and avoiding the use of symbolic operators, resulting in a development experience that is both intuitive and robust.

What is the Kyo?

Drawing inspiration from ZIO's effect rotation, Kyo takes a more generalized approach. While ZIO restricts effects to two channels, dependency injection and short-circuiting, Kyo allows for an arbitrary number of effectful channels. This enhancement gives developers greater flexibility in effect management, while also simplifying Kyo's internal codebase through more principled design patterns.

What Primitives Does Kyo Offer?

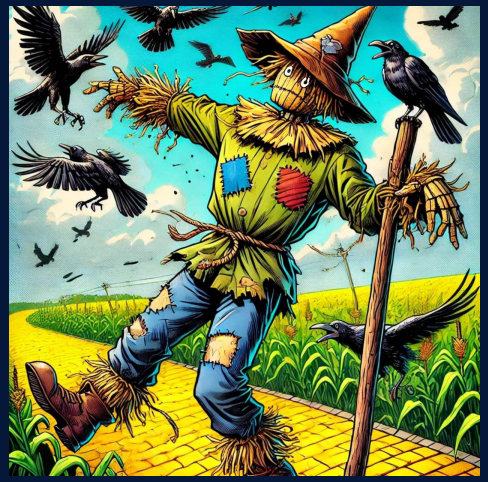
Arrow Effects

The Effects you expect from your MTL style libraries

Contextual Effects

Channels

Unbounded and bounded.



Queues & Hubs

Message Queues and Hubs for broadcasting

And Much More

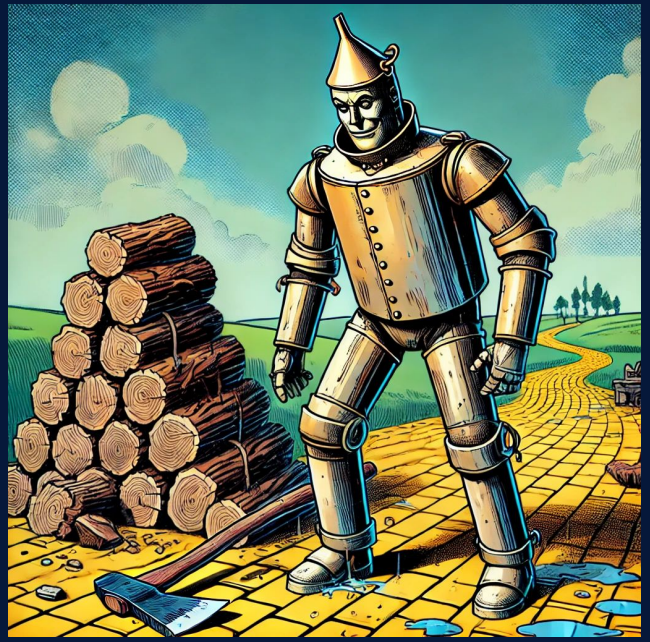
Concurrency Primitives, Console, System, Random, Path, ZIO, Cats, and Tapir Integrations

3. Take Heart

We Have The Technology

Questions?

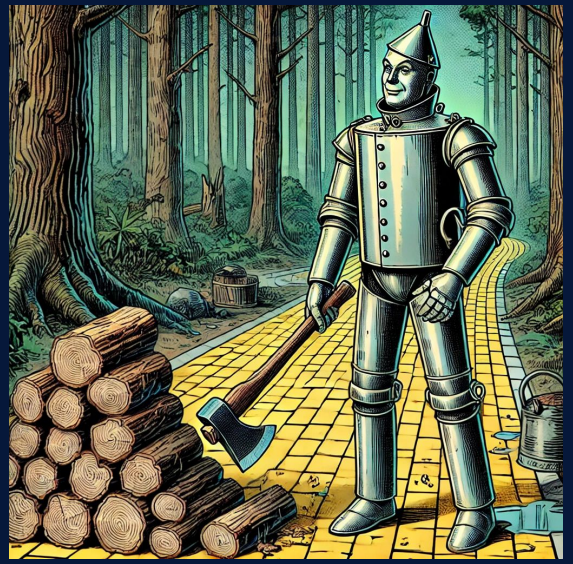
- What Parts Do We Need?
- What Do We Have?
- How Can We Assemble The Parts?



What Do We Need?

What Components Do We Need?

- Actors
- Messages
- Mailbox

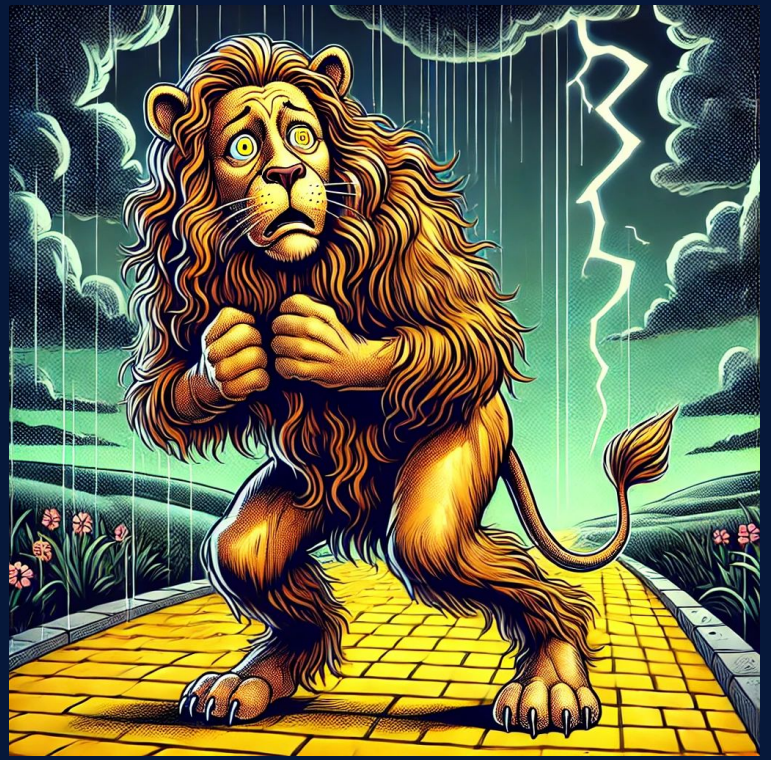


4. Be Brave

We Can Build This

To The Code!

https://github.com/DamianReeves/functional-scala-2024-talk



functional scala 2024

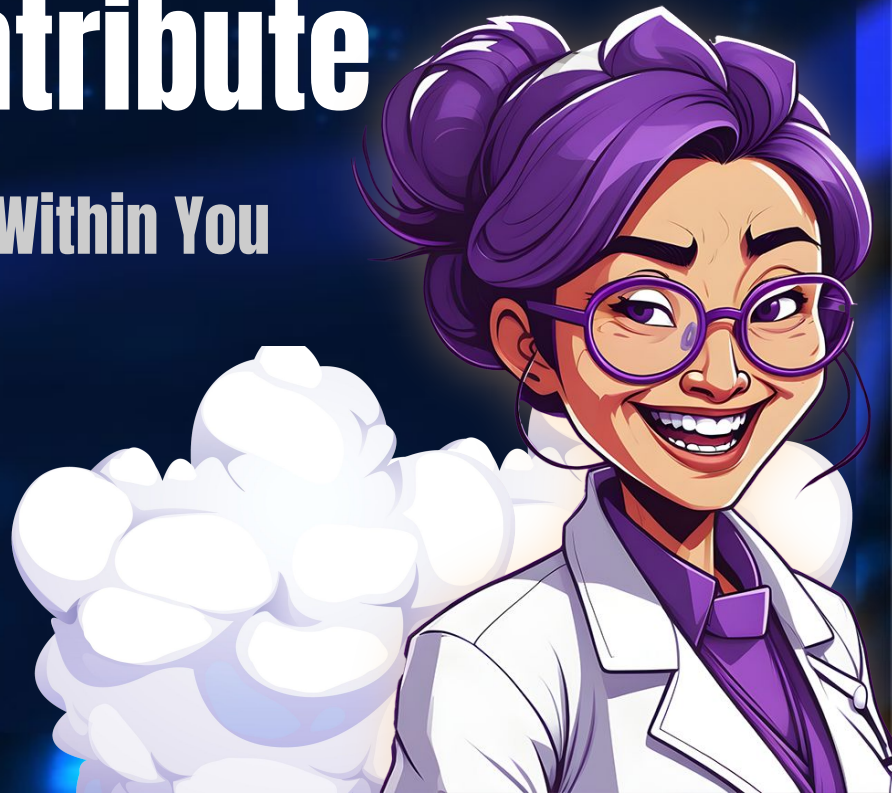
A Call To Contribute

Unlock The Wizard/Mad Scientist Within You

Learn More - https://getkyo.io

Contribute: https://github.com/getkyo/kyo/

KYO



functional scala 2024

Special Thanks

Flavio Brasil

- None of this would be possible without him.

Kyo Community

John DeGoes & Ziverge

Incredible Co-workers @ Capital One

My wife and supportive family

Sponsored by

GOLEM

ZIVERGE

functional scala 2024

THANK YOU

Sponsored by

GOLEM

ZIVERGE