# Getting My Fullstack Back: Elixir and Phoenix.LiveView

Scott Hickey
scott.hickey@protonmail.com

## Speaker Bio

- Professional software engineer for 35 years
  - System rewrites/conversions at FDR/fiServ, United Healthcare, Mutual of Omaha, NNG
  - Brought systems into production after 3-7 previous failures by others teams
- C, C++, Java, Groovy, JavaScript, Elixir (paid)
  - 4GL : Powerbuilder, Clarion, Clipper, dBase, SQL, VX-Rexx
  - JSP, Servlets, Struts, Spring, Micronaut, Vue.js
- Scheme, Lisp, Haskell, Smalltalk, Clojure (non-paid)
- **Elixir**

Past projects are mix of full-stack efforts and large, multi-year multi-million dollar projects.

Previously, Principal Enterprise Architect at Fortune 500 insurance company.

## Agenda

- Intro:
    - Context
    - Case Study Anecdotes
- Glossary / Level Set
- **What is Phoenix LiveView**
- **How Did I Get Here**
- Conclusion

## Context: Large Rewrites, Conversions and Greenfields

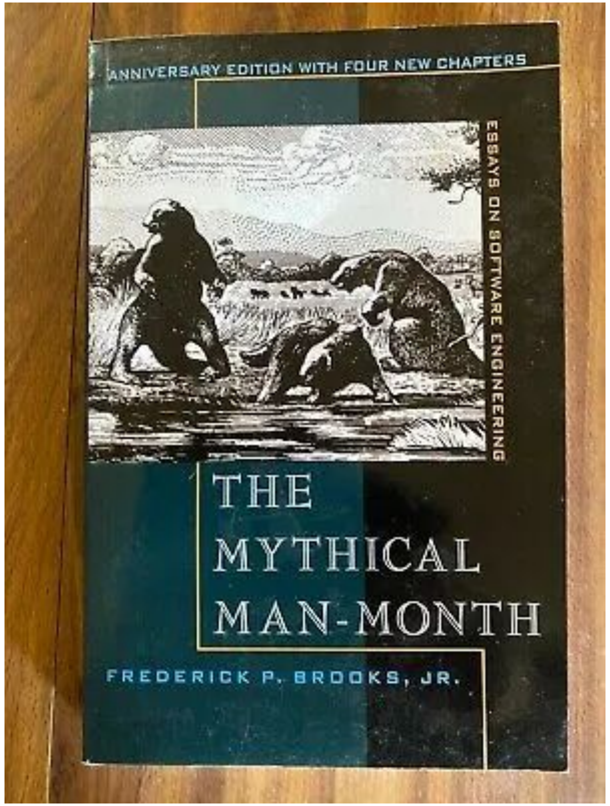
*My* career has mostly been about addressing:

- Expensive Runtimes
- Expensive and Time-consuming Enhancements
- Multi-year backlogs

Multi-year, multi-million dollar projects:

- Hard to bring home
- Should have a shelf-life of longer that 3 years

“The Mythical Man-Month” by Fred Brooks, 1975


ANNIVERSARY EDITION WITH FOUR NEW CHAPTERS
ESSAYS ON SOFTWARE ENGINEERING
THE MYTHICAL MAN-MONTH
FREDERICK P. BROOKS, JR.

# Attention Getting Anecdotes

## Discord (as of 2020)

- 100 million monthly active users
- 12 million concurrent users across all servers
- 26 million WebSocket events to clients per second, and Elixir is powering all of this
- Discord’s chat infrastructure team comprises **five engineers**

## Pintrest (notification system)

- More than 150 million monthly users
- Tenfold drop in the codebase size from **10,000 lines to only 1,000**
- **Cut server requirements by half**

## Attention Getting Anecdotes

- Moz: 63 times less disk space, 20 times faster API
- Financial Times: **easy to learn**, quick to scale
- Bleacher Report
  - "Their development team tried out various options, including Node.js and Go, before finally deciding on Elixir—it brought significant improvements in performance, while its syntax was similar to Ruby's, which made it **easy to learn**."
  - Ability to handle **8x the traffic** without autoscaling
  - Resource intensive features, previously requiring aggressive horizontal scaling, now run on about **1/10th of the servers** with low CPU utilisation
  - 2017, Bleacher Report went from **150 servers to just 5**

## Attention Getting Anecdotes

Bet365 : Worlds Largest Online Betting Site, migrated from Java to Erlang

- Significant reduction in **code complexity**
- Increase in users supported on **a single node from 10s to 100s of thousands**
- Increase in the speed of product development and delivery
- 6 million HTTP requests and more than 500,000 database transactions per second

WhatsApp: 2 million concurrent connections on a single machine

## Why Elixir? I HAD to Take A Closer Look

As an architect or a dev team lead, can’t ignore:

- Reduction in cloud footprint by 95%
- Devs wax rhapsodic about the experience
- Toyota, Pepsico : not just for startups

As someone who’s experience with Groovy and dabbled with Lisp, Scheme, Haskell, Scala, and Clojure – I’m seeing a language that is:

- Practical, not just academic
- No academic baggage
- No JVM Baggage
- VM runtime stability is legendary

## Let’s Define Some Terms

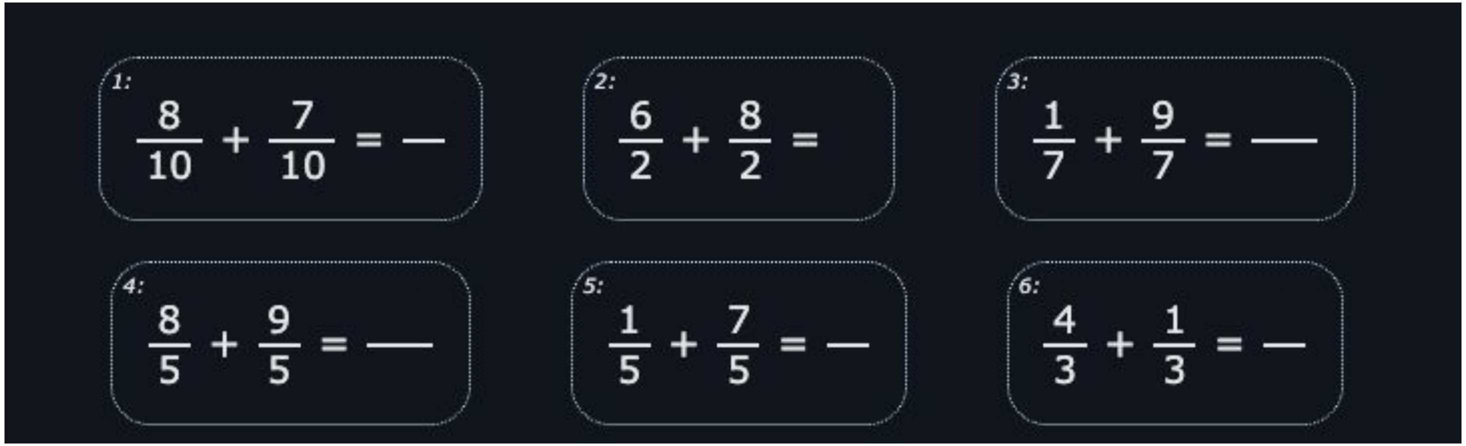
Elixir

- Functional programming language that runs on the Erlang / Beam VM
- Elixir is 10 years old
- Designed to be accessible & concise
- Community expects Ruby on Rails productivity
- Erlang has been around since late 1980’s
- Has legendary reputation for uptime

# Functional Programming

- Higher Order Functions
  - Functions that accept functions as input
  - Functions that return functions
- Pure Functions
  - Don’t reach outside of their scope
  - Don’t mutate anything
- Immutability
  - For the life of a computation, the value won’t change
  - Oracle/Clojure MVCC

## OJUG Presentation circa 2010


1: $\frac{8}{10} + \frac{7}{10} = \underline{\quad}$

2: $\frac{6}{2} + \frac{8}{2} =$

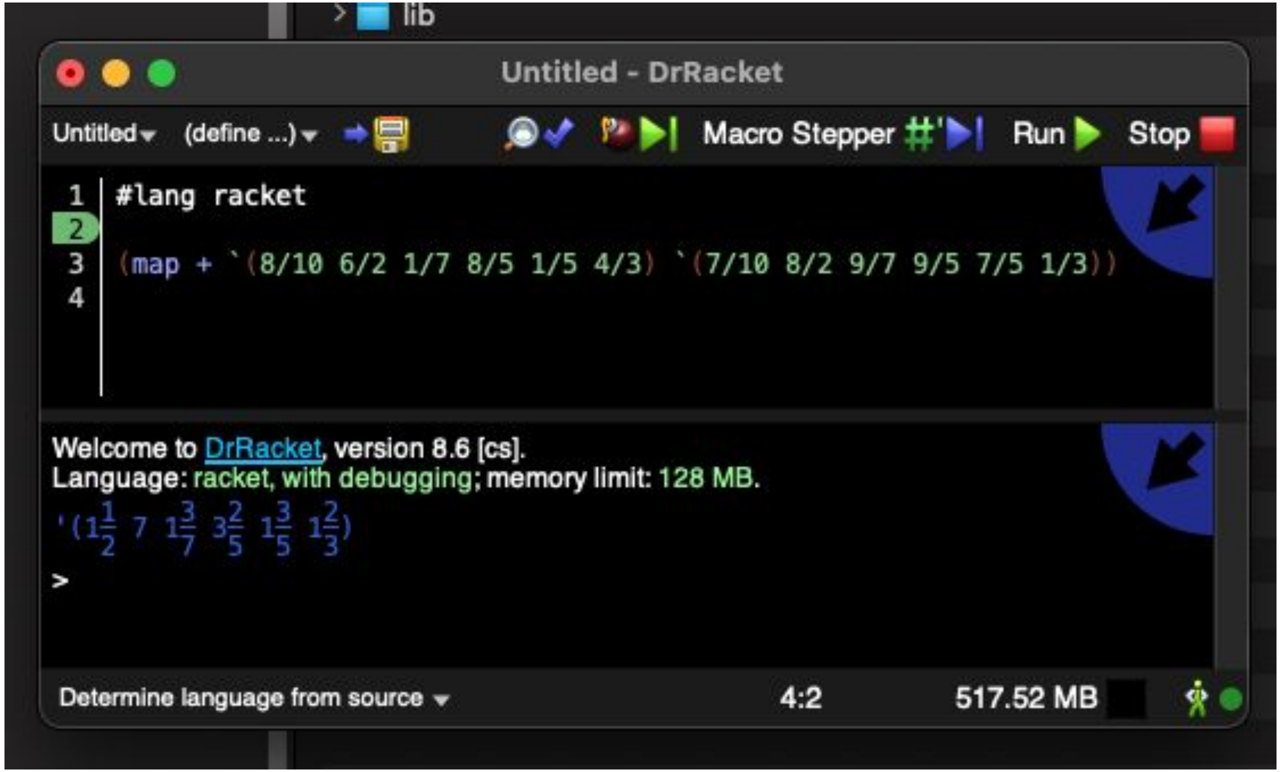
3: $\frac{1}{7} + \frac{9}{7} = \underline{\quad}$

4: $\frac{8}{5} + \frac{9}{5} = \underline{\quad}$

5: $\frac{1}{5} + \frac{7}{5} = \underline{\quad}$

6: $\frac{4}{3} + \frac{1}{3} = \underline{\quad}$

## Six Grader’s Solution: Higher Order Functions



```
#lang racket

(map + `(8/10 6/2 1/7 8/5 1/5 4/3) `(7/10 8/2 9/7 9/5 7/5 1/3))
```

```
Welcome to DrRacket, version 8.6 [cs].
Language: racket, with debugging; memory limit: 128 MB.
'(1 1/2 7 1 3/7 3 2/5 1 3/5 1 2/3)
>
```

# Pure Functions

```
String selectSql = "SELECT * FROM employees";

double total_salaries = 0.0

int counter = 0

try (ResultSet resultSet = stmt.executeQuery(selectSql)) {

    while (resultSet.next()) {

        counter++

        Total_salaries += resultSet.getDouble("salary"));

    }

}

return total_salaries / counter
```

## Pure Function in Groovy

```
// Pure function, could be a list of anything that has a salary
double calcAverageSalary(List employees) {
    employees.collect { emp -> emp.salary } .sum() / employees.size()
}
String selectSql = "SELECT * FROM employees"
// Fetch employees and transform ResultSet into a List of Maps
def employees = sql.rows(selectSql)
```

## Fourth Generation Languages: 4GL

Technopedia

- A fourth generation (programming) language (4GL) is a grouping of programming languages that attempt to get closer than 3GLs to human language, form of thinking and conceptualization.
- 4GLs are designed to reduce the overall time, effort and cost of software development.

Examples: Dbase, FoxPro, Powerbuilder, Progress, SQL, RPG, Scripting Languages e.g. Rexx ( Perl, PHP, Ruby, Python…*maybe sorta*)

*Fourth* GL?

- 1GL : 0110
- 2GL : assembler
- 3GL: C, C++, Java, JavaScript

## 4GL’s Were *Next_Big_Thing* in 1980’s, early 1990’s

*My* Experience

- Built point retail of sale system in Clarion, then wrote in Foxbase in a couple of months
- Built Senior Healthcare Prospect Tracking system using VX-Rexx in a couple of months, with email integration
- Built SAS -> DB/2 ETL system for healthcare claims data warehouse using Rexx

The code reflected the business rules without a lot of ceremony. Low cost to build and stable for years. *There’s something to be said for this.*Then the Web happened.

## Fast Forward to Today

In the last 25 years, we’re better at building software better, faster, cheaper, on-time, on-budget, more stable…

*“By this time next year, 50% of our engineers need to be full-stack.”*

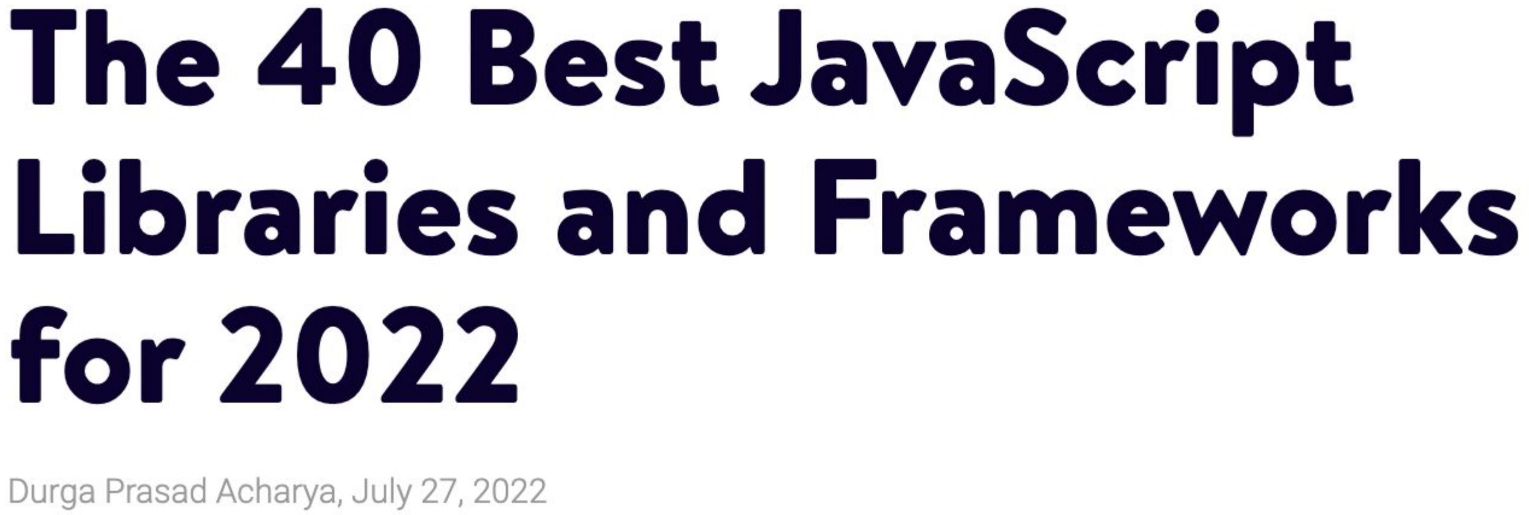
*VP of Software Engineering*

*Full-Stack:*

- Front-end: JavaScript
- Back-end: Java
- Cloud: AWS

Only three things to know - no problem!

## Front-End



The 40 Best JavaScript Libraries and Frameworks for 2022

Durga Prasad Acharya, July 27, 2022

## Front-End

- JavaScript, TypeScript, Flow.js
- WebPack, Gulp, Grunt, Yarn, npm
- React, Vue, Svelte, Ember, Angular
- Async, HTTP requests
- Mustache, handlebars, jsx
- Underscore, Lodash, immutable-js, *other functional programming libraries*
- Routing, State Management, Immutability
- Jest, Jasmine, Mocha, Protractor, Cypress
- MVC, MVVM, MVP
- RX.js, Cycle.js
- Number, String, Date, i18, Logging, Security

## Back-End

- Java 8,9,11,17, 18
- Maven, Gradle
- ORM, SQL, NoSQL,Hibernate, JDBC, jOOQ, Jackson
- Spring, Spring MVC, Spring Boot, Micronaut, Guice
- Checkstyle, PMD, Sonar, Clover, JaCoCo
- JHipster, Lombock
- Apache String, Number, Date, BeanUtils, HTTP, Quartz, Logging, POI, FOP, iText
- Akka, vert.x, RxJava
- JUnit, Spock, JBehave, H2, Geb, JMeter, Gatling

## AWS

- Lambda, Step Functions
- EC2, Containers, Virtual Network
- SQS, SNS
- DynamoDB, S3, RDS
- Cognito
- API Gateway, Kinesis
- ELB
- ElastiCache
- DevOps, Security, IAM, Local run and test, Monitoring

## Seriously, No One Knows It All

In kind of perverse way, we wear mastering the complexity like a badge of honor.

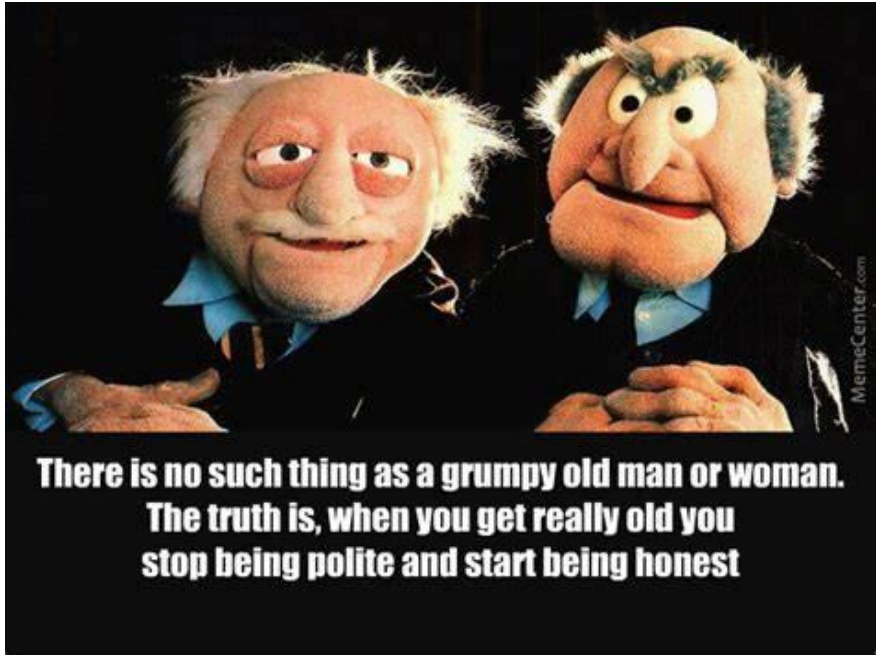
*Sadly, that’s mostly accidental complexity.*

The previous three slides haven’t each touched **Domain Knowledge.**

**=> Too Much Stuff**

## There Has To Be a Better Way

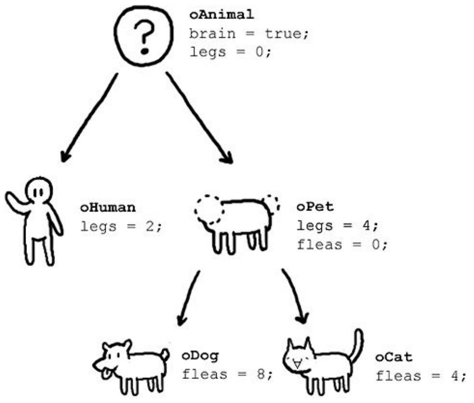
- The barrier for entry didn't use to be this high.
- Java/JavaScript are 25 years old; nothing better since? Really??
- "We just rewrite it every three years."
- "I made one change and it took three weeks to track down the bug."
- "I need a week to get our new project development environment setup before our team can start working on it." - is this a quote from 2002 or 2022?


MemeCenter.com
There is no such thing as a grumpy old man or woman.
The truth is, when you get really old you
stop being polite and start being honest

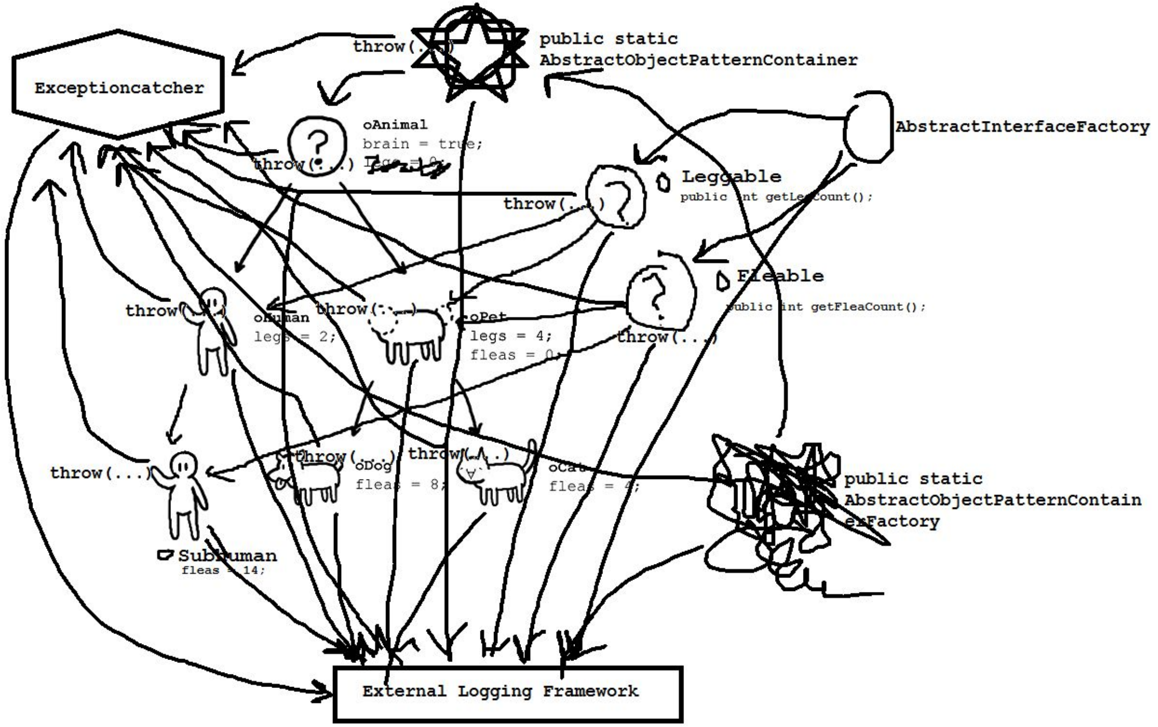
## But at Least Our Programming Language is Great

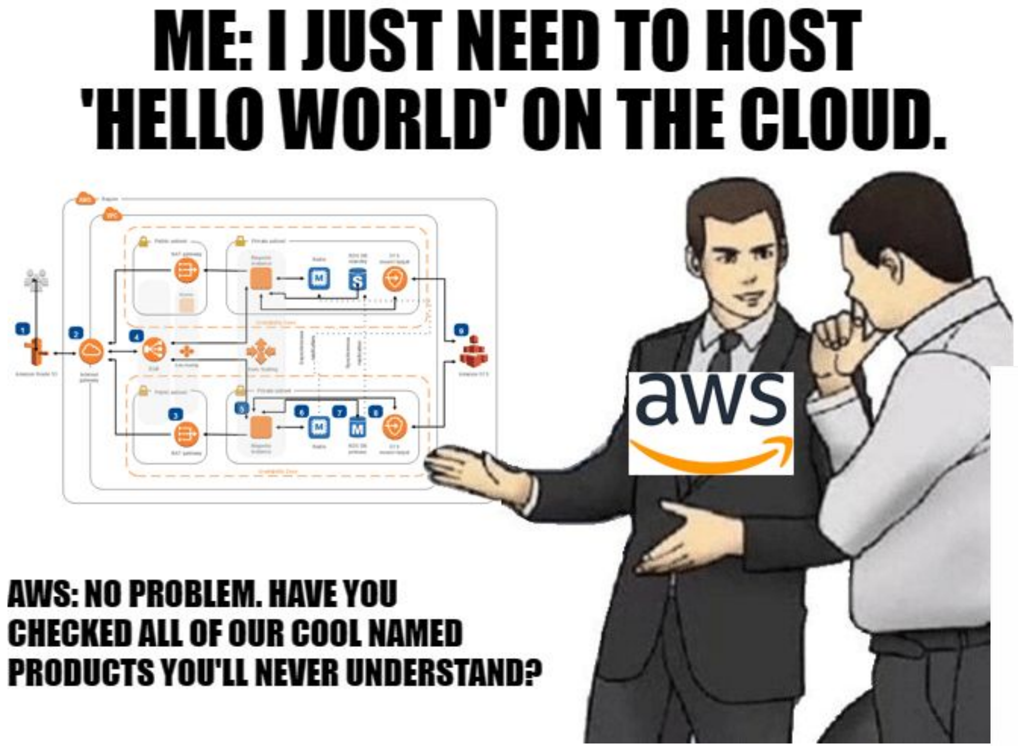
Debugging in Javascript, be like…

## What OOP users claim



## What actually happens




ME: I JUST NEED TO HOST 'HELLO WORLD' ON THE CLOUD.
aws
AWS: NO PROBLEM. HAVE YOU CHECKED ALL OF OUR COOL NAMED PRODUCTS YOU'LL NEVER UNDERSTAND?

BRACE YOURSELVES
THE AWS BILL IS COMING
imgflip.com

## “Do I see any impediment that prevents me or the development team from meeting the sprint goal?”

Essential Complexity

- Domain
- Responsive UI
- Data storage and access

Accidental Complexity

- Frameworks
- Languages
    - Mutability
    - Side-effects
- Auto-gen ceremony
- Libraries, Transitive Dependencies

## Finally: LiveView

Beam: The Virtual Machine

- Beam -> JVM
- Distributed cache, pub-sub, thread safety, lightweight threads, distributed

Elixir: The Programming Language

- Erlang -> Java, Groovy -> Elixir

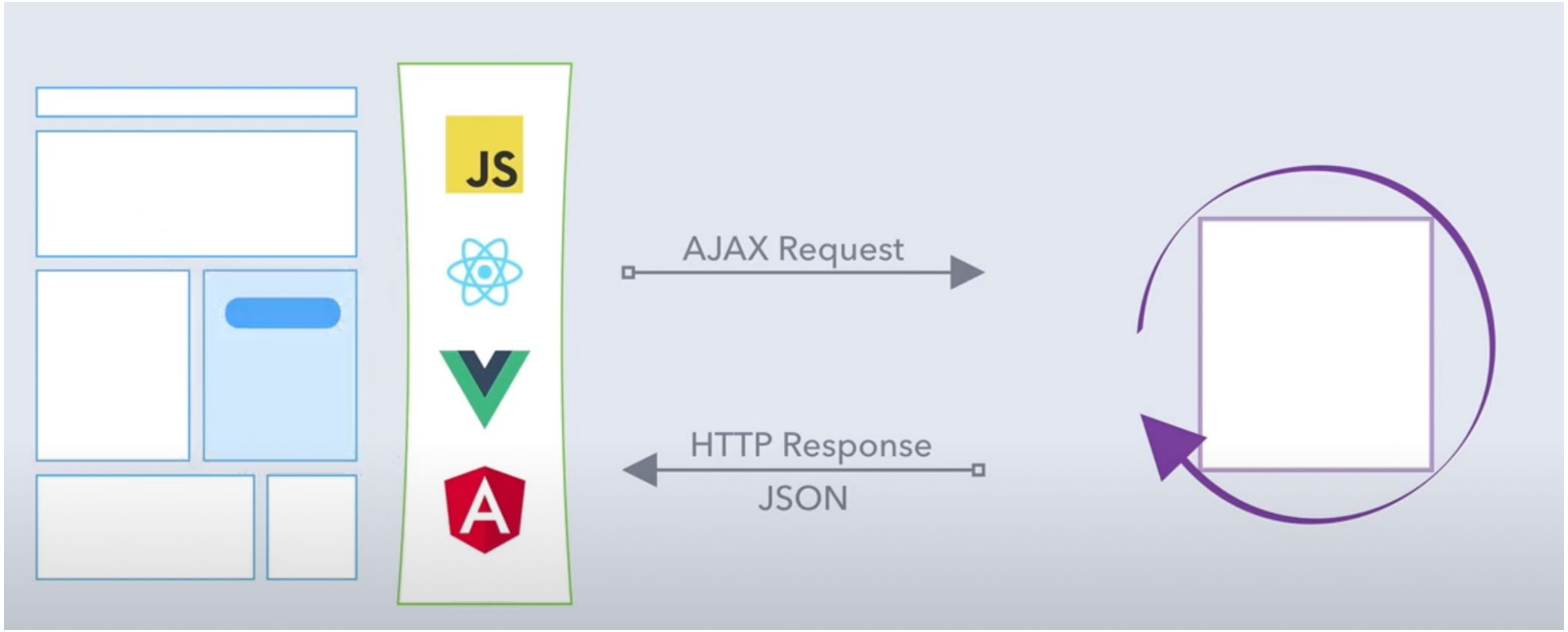
Phoenix: The Web Framework

- Squinting, Spring MVC, Rails, Grails

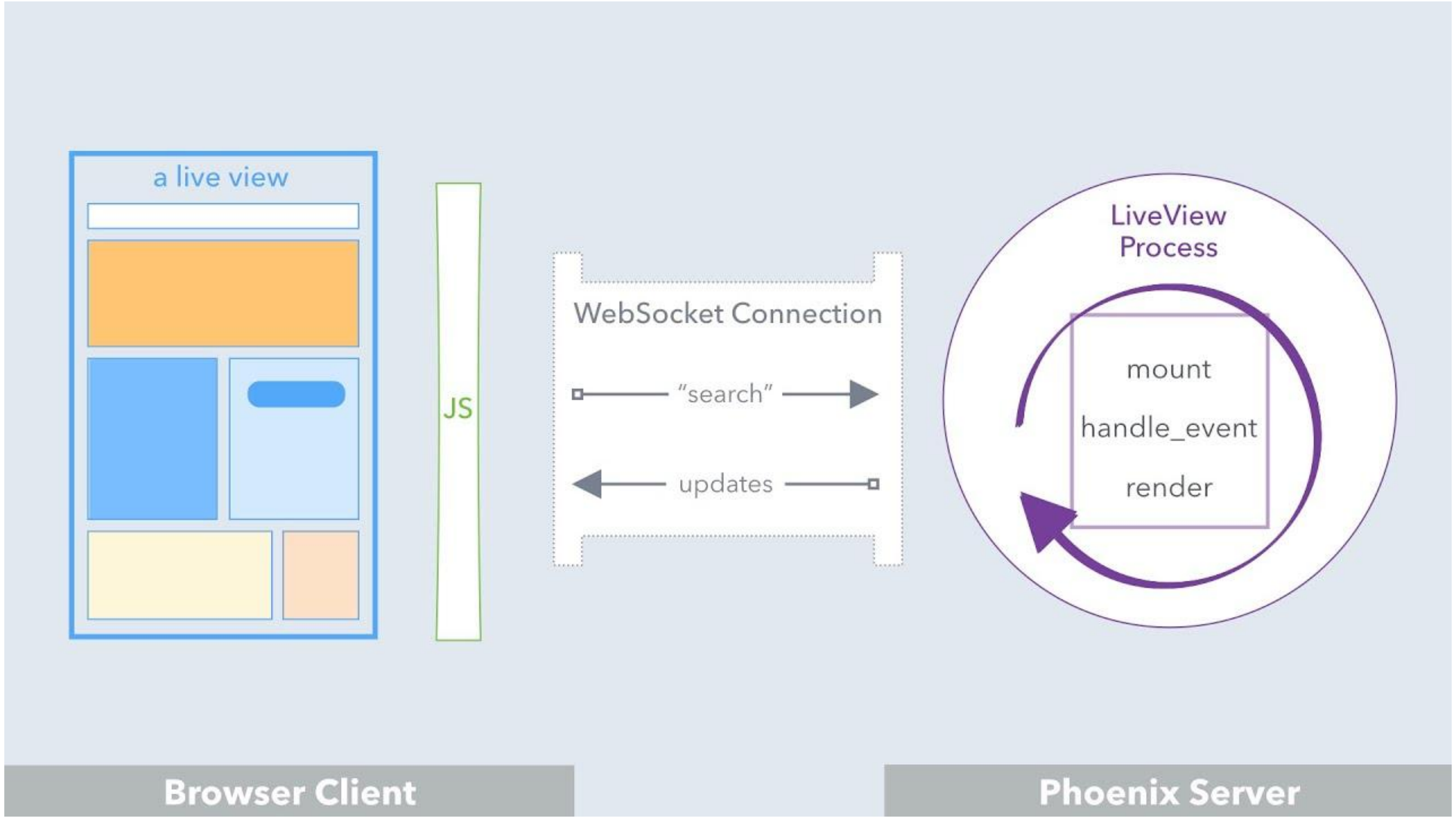
LiveView: Part of Phoenix

- ***Like nothing you’ve ever seen***

# Modern Web Development



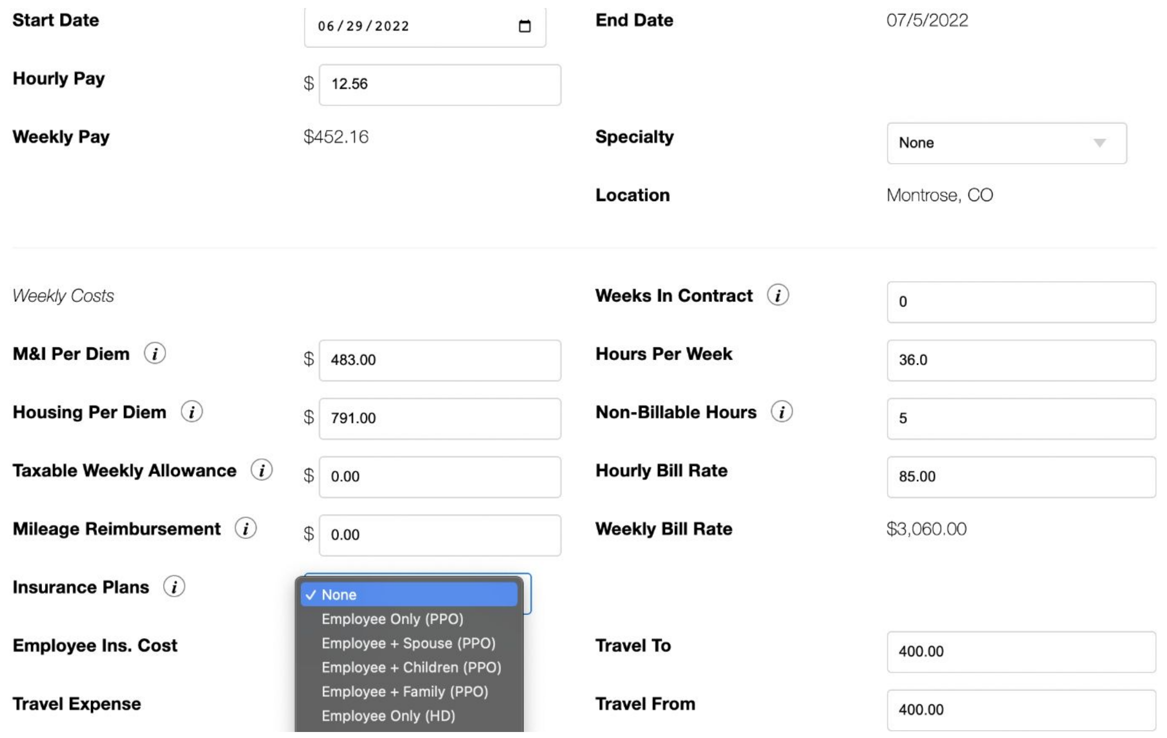
## Phoenix.LiveView



## Server Side….

- State management
- Routing
- Event Handling
- Data / UI Template Render
- DOM diff
- API calls

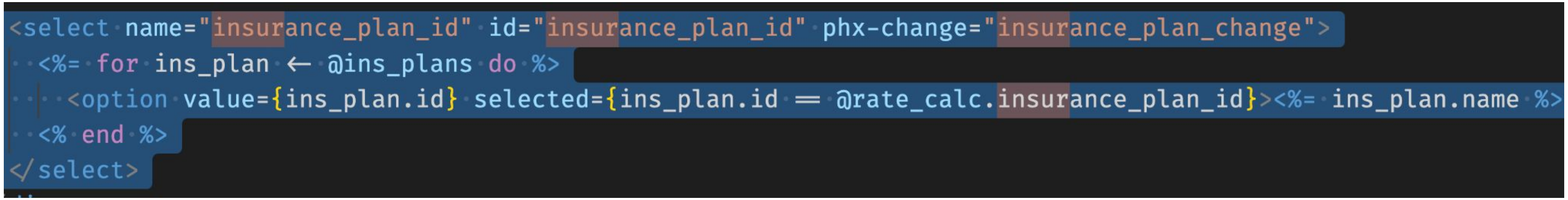
# Application Screenshot



| | | | |
|---|---|---|---|
| **Start Date** | 06/29/2022 | **End Date** | 07/5/2022 |
| **Hourly Pay** | $ 12.56 | | |
| **Weekly Pay** | $452.16 | **Specialty** | None |
| | | **Location** | Montrose, CO |

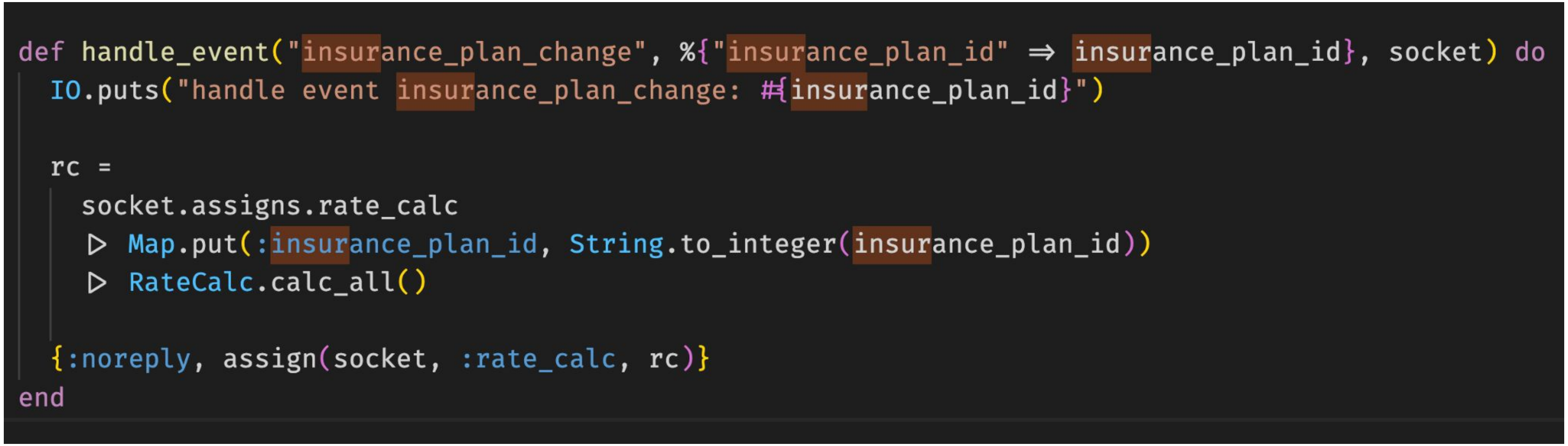
| | | | |
|---|---|---|---|
| *Weekly Costs* | | **Weeks In Contract** | 0 |
| **M&I Per Diem** | $ 483.00 | **Hours Per Week** | 36.0 |
| **Housing Per Diem** | $ 791.00 | **Non-Billable Hours** | 5 |
| **Taxable Weekly Allowance** | $ 0.00 | **Hourly Bill Rate** | 85.00 |
| **Mileage Reimbursement** | $ 0.00 | **Weekly Bill Rate** | $3,060.00 |
| **Insurance Plans** | ✓ None | | |
| **Employee Ins. Cost** | | **Travel To** | 400.00 |
| **Travel Expense** | | **Travel From** | 400.00 |

# UI Template



```
<select name="insurance_plan_id" id="insurance_plan_id" phx-change="insurance_plan_change">
  <%= for ins_plan <- @ins_plans do %>
    <option value={ins_plan.id} selected={ins_plan.id == @rate_calc.insurance_plan_id}><%= ins_plan.name %>
  <% end %>
</select>
```

## Event Handler



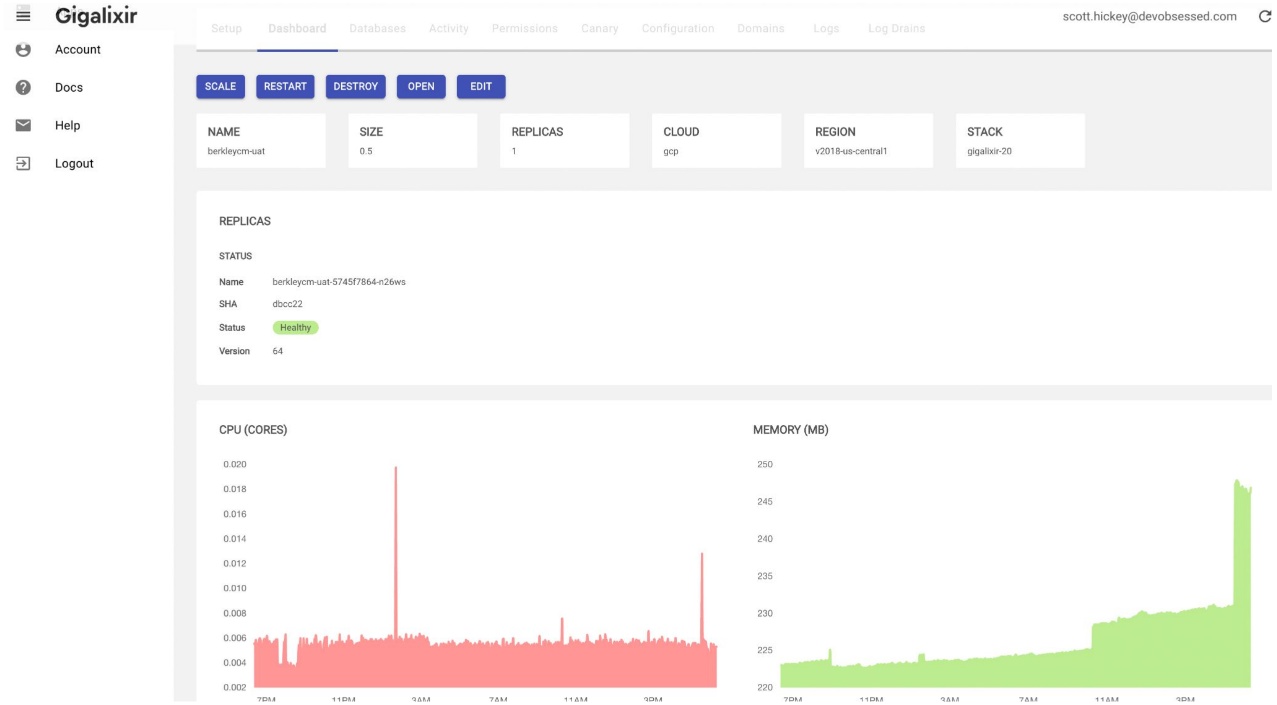
```elixir
def handle_event("insurance_plan_change", %{"insurance_plan_id" ⇒ insurance_plan_id}, socket) do
  IO.puts("handle event insurance_plan_change: #{insurance_plan_id}")

  rc =
    socket.assigns.rate_calc
    ▷ Map.put(:insurance_plan_id, String.to_integer(insurance_plan_id))
    ▷ RateCalc.calc_all()

  {:noreply, assign(socket, :rate_calc, rc)}
end
```

# Less Stuff: Deployment

Heroku, Fly.io, Gigalixor -> No certifications required!



| NAME | SIZE | REPLICAS | CLOUD | REGION | STACK |
|---|---|---|---|---|---|
| berkleycm-uat | 0.5 | 1 | gcp | v2018-us-central1 | gigalixir-20 |

REPLICAS

STATUS

| | |
|---|---|
| Name | berkleycm-uat-5745f7864-n26ws |
| SHA | dbcc22 |
| Status | Healthy |
| Version | 64 |

## Agenda

- Intro
- Phoenix LiveView
- **How Did I Get Here**
- Conclusion

## All Java Books to Goodwill? Really??

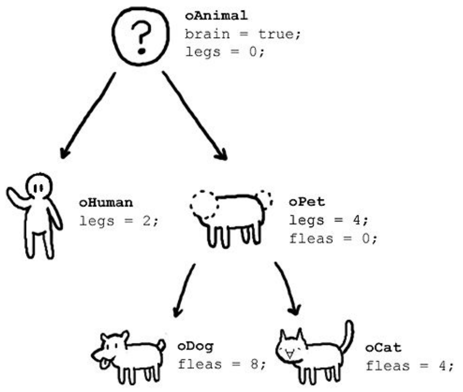
*“A language that doesn't affect the way you think about programming, is not worth knowing.”*

*Alan Perlis*

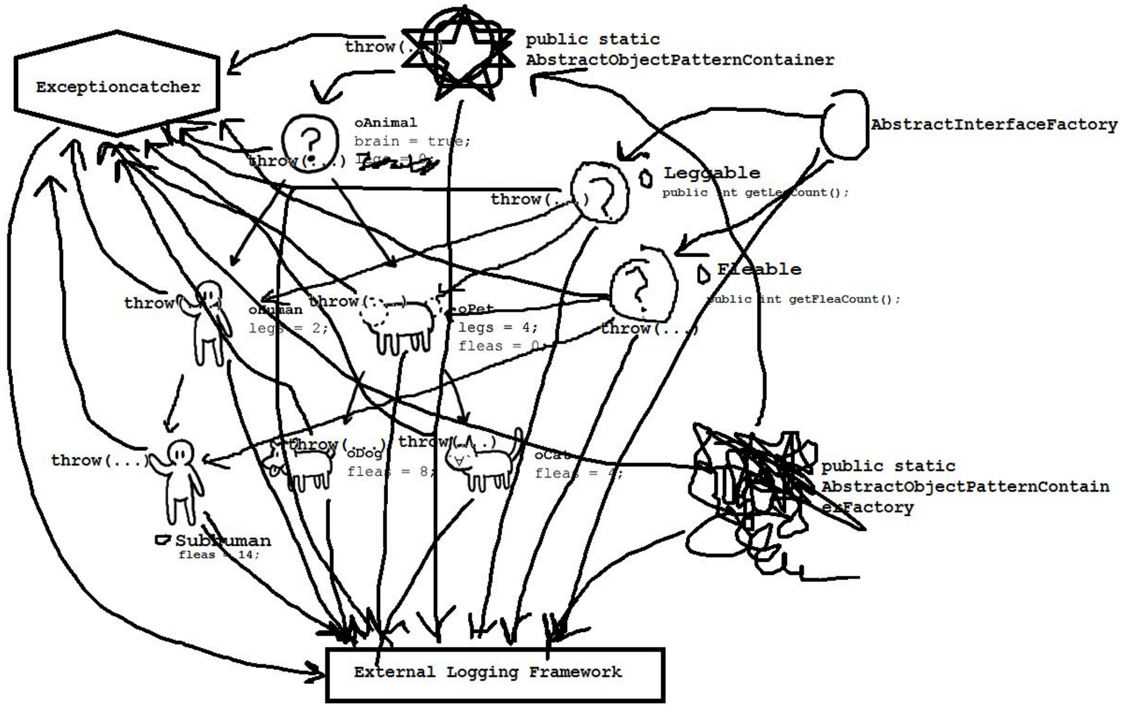
I was ready to move on from:

- Bolt-on Functional Programming
  - Groovy was really great, if everyone follows functional conventions
  - Really *wanted* to love it
  - Kotlin looks great coming from Java, not-so-great coming from Groovy
  - Scala is a mess
- Over-Engineered Solutions
  - Frameworks
  - Libraries
  - Cloud architectures

## What OOP users claim



## What actually happens



## Large Scale O-O Feels Like A Rube-Goldberg Machine

# Functional Programming Big Wins

- Programmer Joy - Literally
  - Smaller functions, less cognitive load *over time*
  - No Mocks
  - No State
  - Easier to Manage
- Huge impact on bringing home large projects
  - Several multi-year, multi-million dollar projects
  - Projects didn't squirrel sideways mid-project
  - No Long Tail

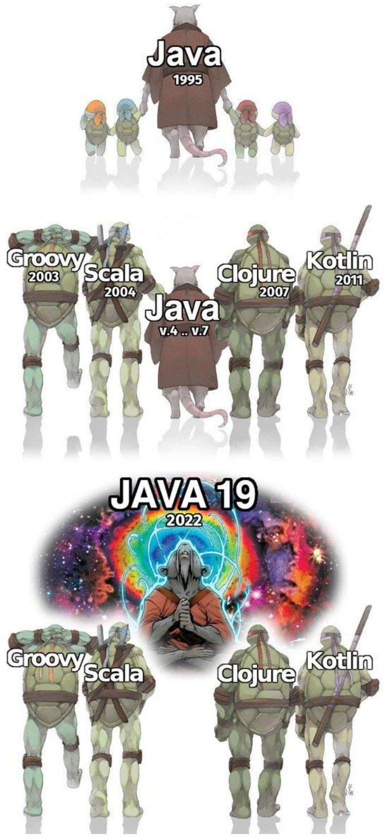
"It is better to have 100 functions operate on one data structure than 10 functions on 10 data structures."

— **Alan Perlis**

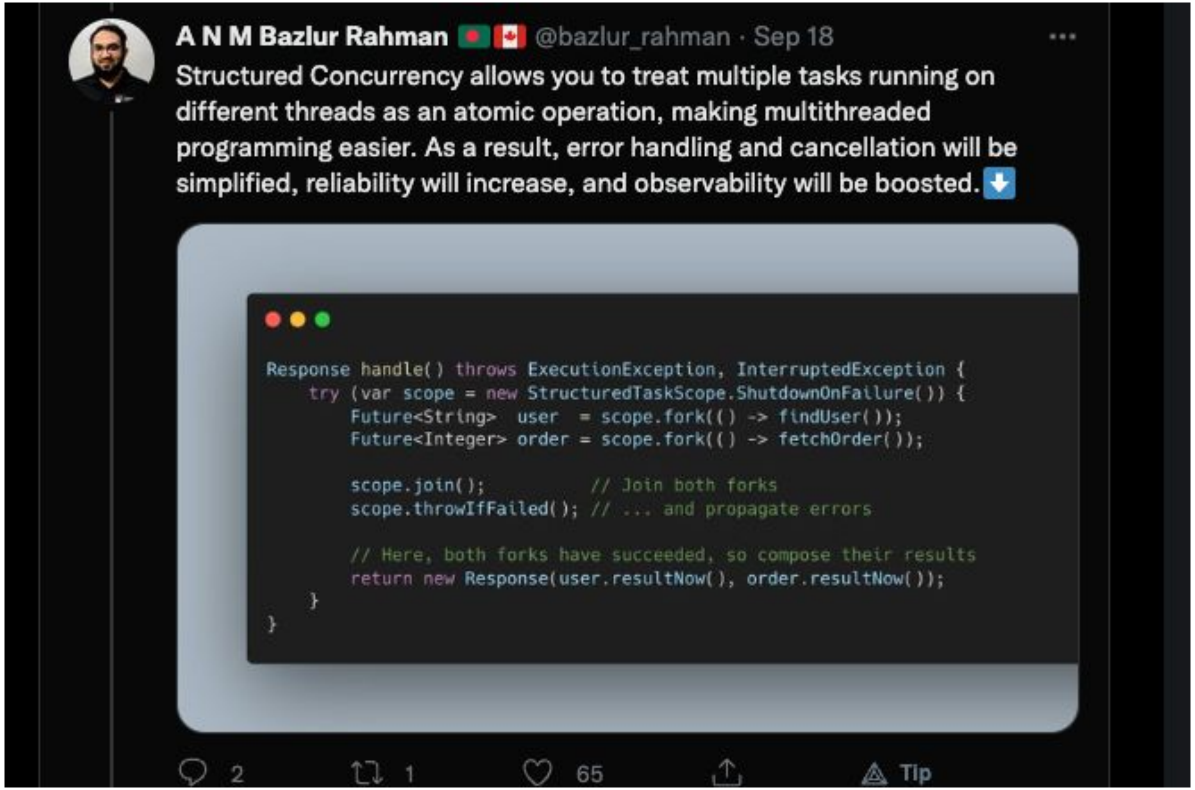
"All race conditions, deadlock conditions, and concurrent update problems are due to mutable variables."

— **Robert C. Martin, Clean Architecture**

## Bolt-on


Java
1995
Groovy
2003
Scala
2004
Java
v.4 .. v.7
Clojure
2007
Kotlin
2011
JAVA 19
2022
Groovy
Scala
Clojure
Kotlin

## Sounds Like Erlang From 30 Years Ago


A N M Bazlur Rahman 🇧🇩 🇨🇦 @bazlur_rahman · Sep 18

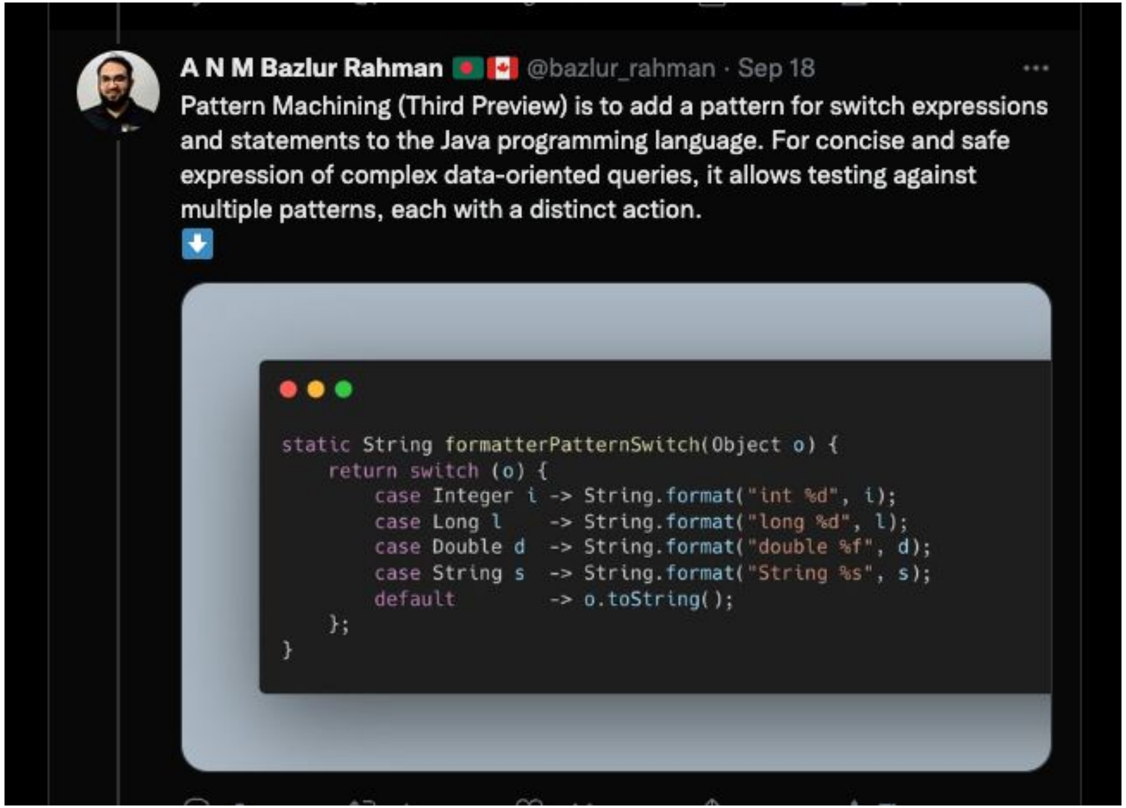
Structured Concurrency allows you to treat multiple tasks running on different threads as an atomic operation, making multithreaded programming easier. As a result, error handling and cancellation will be simplified, reliability will increase, and observability will be boosted. ⬇️

```
Response handle() throws ExecutionException, InterruptedException {
    try (var scope = new StructuredTaskScope.ShutdownOnFailure()) {
        Future<String>  user  = scope.fork(() -> findUser());
        Future<Integer> order = scope.fork(() -> fetchOrder());

        scope.join();          // Join both forks
        scope.throwIfFailed(); // ... and propagate errors

        // Here, both forks have succeeded, so compose their results
        return new Response(user.resultNow(), order.resultNow());
    }
}
```

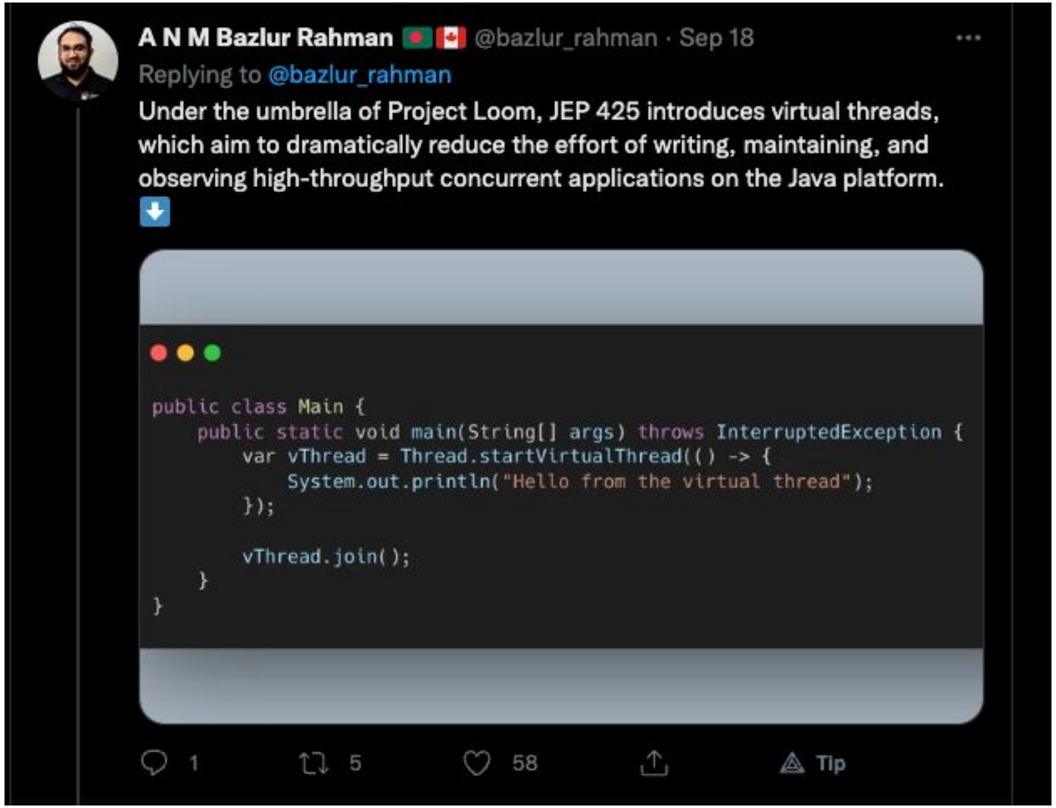
## Sounds Like Erlang & Haskell From 30 Years Ago…


A N M Bazlur Rahman 🇧🇩🇨🇦 @bazlur_rahman · Sep 18

Pattern Machining (Third Preview) is to add a pattern for switch expressions and statements to the Java programming language. For concise and safe expression of complex data-oriented queries, it allows testing against multiple patterns, each with a distinct action.
⬇️

```
static String formatterPatternSwitch(Object o) {
    return switch (o) {
        case Integer i -> String.format("int %d", i);
        case Long l    -> String.format("long %d", l);
        case Double d  -> String.format("double %f", d);
        case String s  -> String.format("String %s", s);
        default        -> o.toString();
    };
}
```

## Sounds Like Erlang From 30 Years Ago



A N M Bazlur Rahman 🇧🇩🇨🇦 @bazlur_rahman · Sep 18

Replying to @bazlur_rahman

Under the umbrella of Project Loom, JEP 425 introduces virtual threads, which aim to dramatically reduce the effort of writing, maintaining, and observing high-throughput concurrent applications on the Java platform. ⬇️

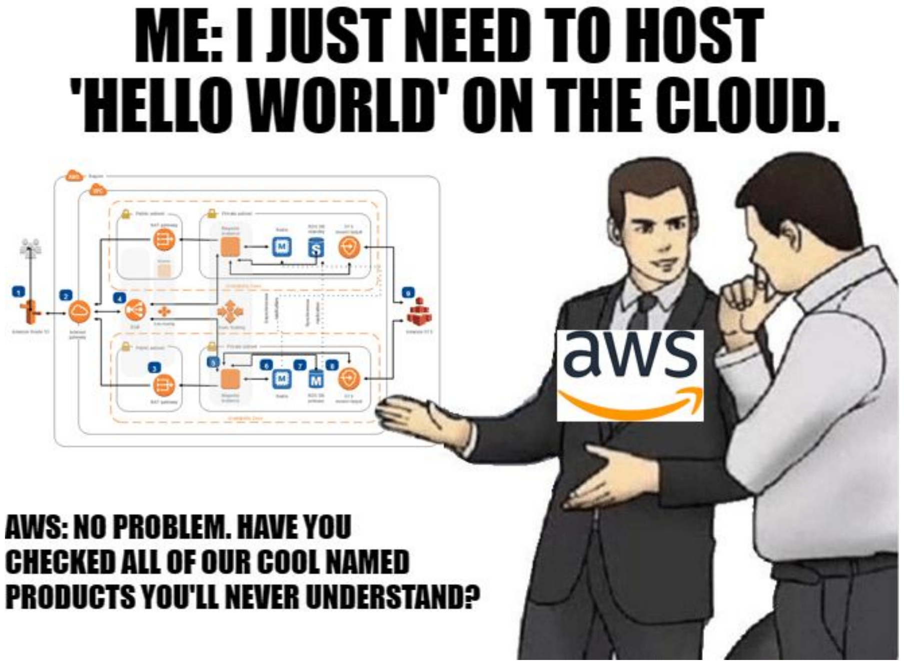
```java
public class Main {
    public static void main(String[] args) throws InterruptedException {
        var vThread = Thread.startVirtualThread(() -> {
            System.out.println("Hello from the virtual thread");
        });

        vThread.join();
    }
}
```

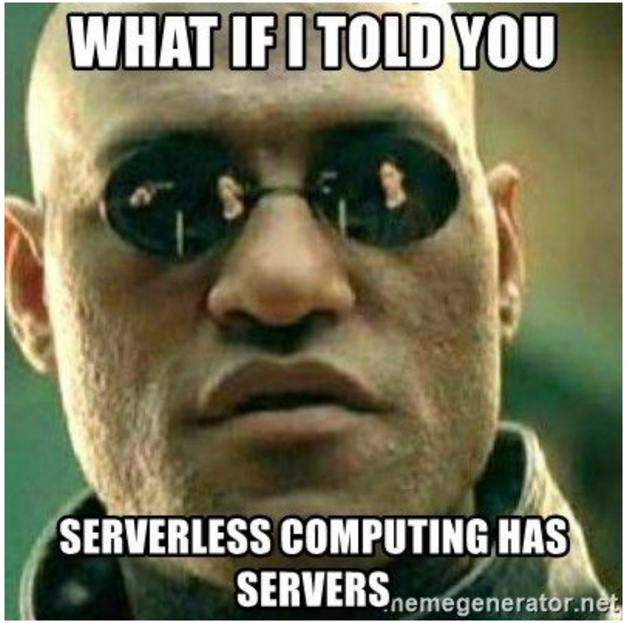
1 5 58 Tip

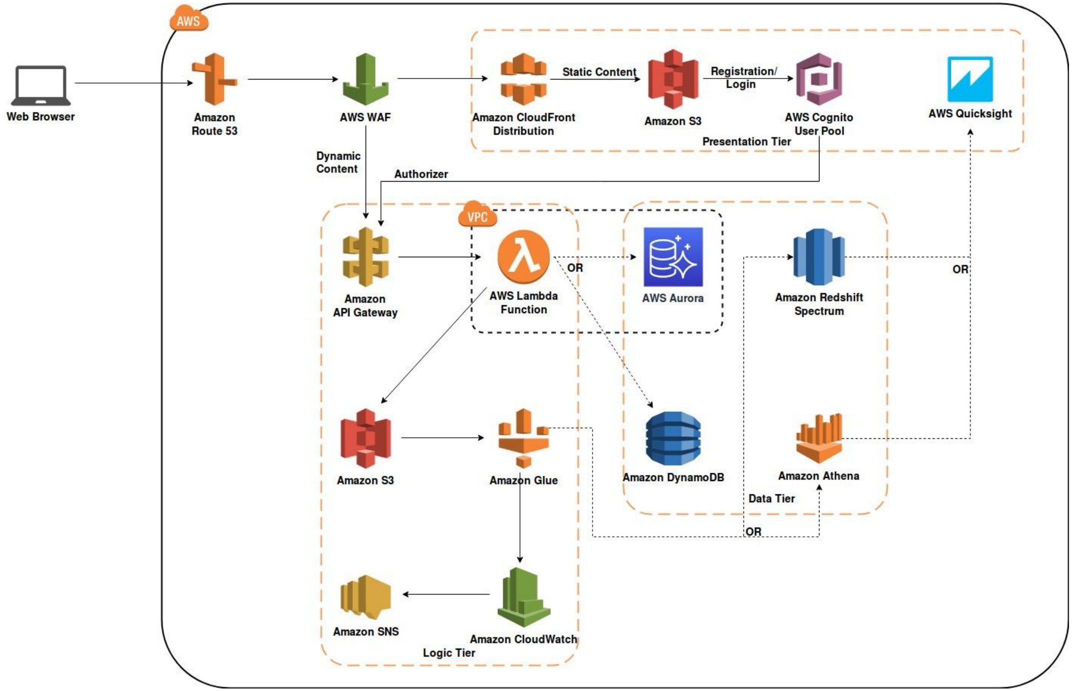
## Over-Engineered Solutions

We still mostly build database backed web-apps with some API calls, pub-sub and caching thrown in.


ME: I JUST NEED TO HOST 'HELLO WORLD' ON THE CLOUD.
aws
AWS: NO PROBLEM. HAVE YOU CHECKED ALL OF OUR COOL NAMED PRODUCTS YOU'LL NEVER UNDERSTAND?

## Serverless…


WHAT IF I TOLD YOU
SERVERLESS COMPUTING HAS SERVERS
memegenerator.net


AWS
Web Browser
Amazon Route 53
AWS WAF
Amazon CloudFront Distribution
Static Content
Amazon S3
Registration/ Login
AWS Cognito User Pool
AWS Quicksight
Presentation Tier
Dynamic Content
Authorizer
VPC
Amazon API Gateway
AWS Lambda Function
OR
AWS Aurora
Amazon Redshift Spectrum
OR
Amazon S3
Amazon Glue
Amazon DynamoDB
Amazon Athena
Data Tier
OR
Amazon SNS
Amazon CloudWatch
Logic Tier

## The BEAM VM: Cloud in-a-box

- One machine scales to 1,000,000 processes
- Built-in hypervisor
- Built-in light-weight messaging
- Built-in Distributed cache
- Per-thread Garbage Collection
- Unparalleled uptime
- Less stuff in the cloud: scale vertically first, let it do what it does best

## Conclusion: Visualize Your Future

- Culture of “Front End”, “Back End”, “Infrastructure” teams
- “*My* code is ready, I’m just waiting on *them*.”
- What’s cooler?
  - “We have a multi-repo” or “We don’t need a multi-repo”
  - “We have hundreds of dependencies” vs “We have 10’s of dependencies”
  - “Here’s a link to my doc for setting up, running and testing our hosted app on your laptop.” vs “just run it”
- Putting your “badges of honor” for mastering complexity on a shelf, never to be seen again.

## Seriously - Change *My* Framework?

Ruby on Rails

Spring MVC

React/Vue.js

*You were using something else when you found your current one.*

## Getting Started

Using functional programming techniques in your primary language is a good place to start.

- Write code without instance variables
- Think in terms of set transformations
- Eliminate mutation by convention

Doing this guarantees:

- You’ll be thrilled with a better code base
- You’ll be frustrated that your language, compile, runtime doesn’t do more

# Hands On Path to Full-Stack Enlightenment

Elixir is a functional programing language. If you don't know functional programing:

- "Little Schemer" + Racket (Scheme)

Learn Elixir:

- Book: "Programming Elixir" by Dave Thomas
- **Video Course: "Elixir for Programmers" by Dave Thomas**
  - Not comprehensive, but retrains your brain
  - Covers Phoenix and LiveView
- Book: "Programming Phoenix LiveView" by Bruce A. Tate and Sophie DeBenedetto
- HumbleBundle!!

## Questions?