Elixir for Rubyists

as a great alternative for web APIs

Elixir’s history

How did it all start?

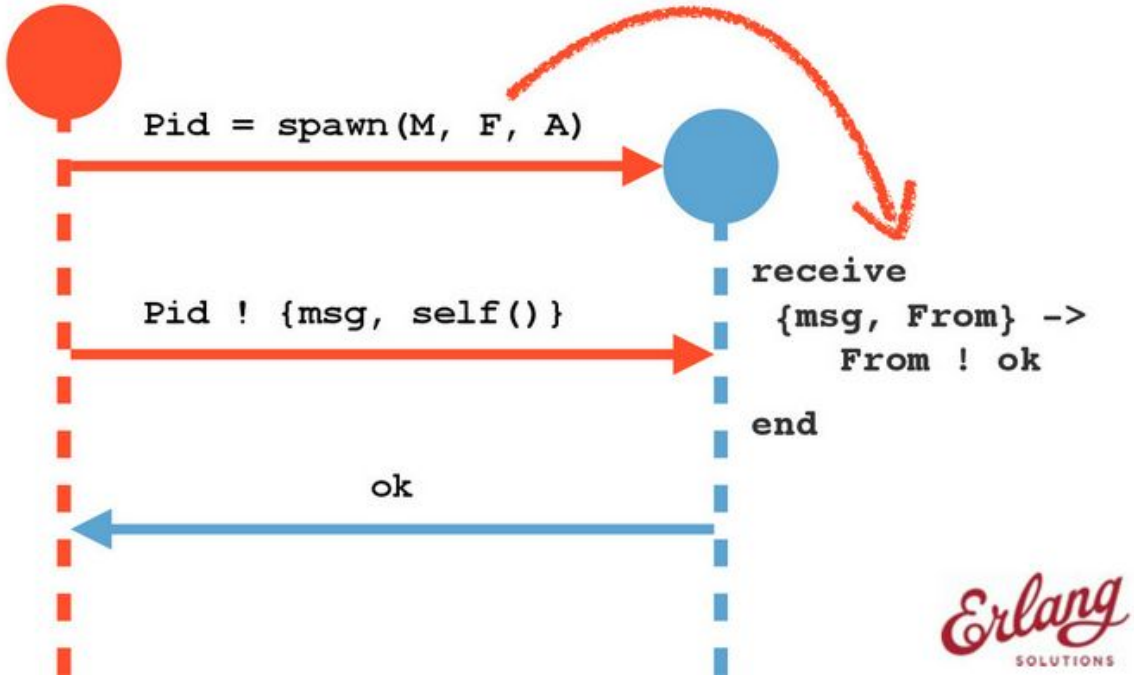
History of Erlang

- Started out in Ericsson software lab in 1987
- Released as open source in 1998
- SMP support as of 2005

Erlang properties

- Functional programming language
- Prolog-like syntax
- Lightweight, massive concurrency
- Asynchronous communication
- Process isolation (& garbage collection)
- Soft real-time

Erlang properties - processes



(image from https://speakerdeck.com/michalslaski/introduction-to-erlang)

Erlang’s VM: BEAM

- runs as one OS process
- process virtual machine
  - creates its’ own “Erlang processes”
- fast and concurrent - systems can run millions of Erlang processes at once
- examples: WhatsApp, CouchDB, Riak database, RabbitMQ
- constantly improving

History of Elixir

- Created in 2011 by José Valim
  - co-founder of Plataformatec
  - member of rails-core
  - you might know him from gems like: inherited_resources, devise, simple_form, responders
- 1.0 stable as of September 2014

Elixir as a language

Language features; syntax

Elixir properties

- Built on top of Erlang
- Compiles to bytecode for the Erlang VM
- Ruby-like syntax
- Additional features, not available in Erlang, easing the developer’s flow:
  - Metaprogramming
  - Pipeline operator
  - Polymorphism via protocol

Elixir syntax

http://learnxinyminutes.com/docs/elixir/

Elixir features

1. Functional programming
    a. Immutable data
    b. Lists and Recursion
    c. Pattern matching
2. Enumerables & Streams
3. Meta-programming
    a. Macros
    b. DSL

Code sample

```elixir
defmodule Company do
  def new(name), do: {name, []}

  def add_employee({company_name, employees}, new_employee) do
    {company_name, [new_employee | employees]}
  end
end

company_1 = Company.new("Initech")
company_2 = Company.add_employee(company_1, "Peter Gibbons")
company_3 = Company.add_employee(company_2, "Michael Bolton")
```

Code sample

```elixir
# 1. Map over an array using Enumerable protocol
Enum.map([1, 2, 3], fn x -> x * 2 end)
# => [2, 4, 6]

# 2. Multiple Enum functions
Enum.sum(Enum.filter(Enum.map(1..100_000, &(&1 * 3)), odd?))
# => 7500000000

# 3. Multiple Enum functions using pipeline |> operator
1..100_000 |> Enum.map(&(&1 * 3)) |> Enum.filter(odd?) |> Enum.sum
# => 7500000000

# 4. Same as above, but using Streams
1..100_000 |> Stream.map(&(&1 * 3)) |> Stream.filter(odd?) |> Enum.sum
# => 7500000000
```

Code sample

```
# 5. Enums are eager, Streams are lazy:
# Here's the difference:
1..3 |>
  Enum.map(&IO.inspect(&1)) |>
  Enum.map(&(&1 * 2)) |>
  Enum.map(&IO.inspect(&1))
1
2
3
2
4
6
#=> [2,4,6]
```

```
stream = 1..3 |>
  Stream.map(&IO.inspect(&1)) |>
  Stream.map(&(&1 * 2)) |>
  Stream.map(&IO.inspect(&1))
Enum.to_list(stream)
1
2
2
4
3
6
#=> [2,4,6]
```

Elixir ecosystem

Because without tools you'll need to do everything from scratch

Elixir ecosystem

1. Hex
2. Phoenix
3. Ecto
4. ExUnit
5. Hound
6. https://github.com/h4cc/awesome-elixir

Phoenix Framework

1. Structure like in rails!
2. Templates
3. Pipelines
4. Websockets integrated
5. All Erlang & Elixir benefits

Phoenix - controller

```elixir
defmodule Audashboard.Api.TaskController do
  use Audashboard.Web, :controller

  alias Audashboard.Task

  plug :scrub_params, "task" when action in [:create, :update]
  plug :action

  def index(conn, _params) do
    tasks = Repo.all(Task)
    render(conn, "index.json", tasks: tasks)
  end

  def create(conn, %{"task" => task_params}) do
    changeset = Task.changeset(%Task{}, task_params)

    if changeset.valid? do
      task = Repo.insert(changeset)
      render(conn, "show.json", task: task)
    else
      conn
      |> put_status(:unprocessable_entity)
      |> render(Audashboard.ChangesetView, "error.json", changeset: changeset)
    end
  end
```

Phoenix - sockets

```elixir
defmodule HelloPhoenix.RoomChannel do
  use Phoenix.Channel

  def join("rooms:lobby", auth_msg, socket) do
    {:ok, socket}
  end
  def join("rooms:" <> _private_room_id, _auth_msg, socket) do
    {:error, %{reason: "unauthorized"}}
  end

  def handle_in("new_msg", %{"body" => body}, socket) do
    broadcast! socket, "new_msg", %{body: body}
    {:noreply, socket}
  end

  def handle_out("new_msg", payload, socket) do
    push socket, "new_msg", payload
    {:noreply, socket}
  end
end
```

Phoenix performance

Framework	Throughput (req/s)	Latency (ms)	Consistency (σ ms)
Gin	51483.20	1.94	0.63
Phoenix	43063.45	2.82	*(1)* 7.46
Express Cluster	27669.46	3.73	2.12
Martini	14798.46	6.81	10.34
Sinatra	9182.86	6.55	3.03
Express	9965.56	10.07	0.95
Rails	3274.81	17.25	6.88
Plug *(1)*	54948.14	3.83	12.40
Play *(2)*	63256.20	1.62	2.96

iMac Intel Core i7 (4.0 GHz 4 Cores) 32 GB RAM

(1) Consistency for both Erlang solutions have become more unstable in this round of tests compared to previous

Elixir as a whole

Is it worth it?

Elixir - is it worth it? (sum up)

- to what things elixir is good for?
- ecosystem, community, libraries are evolving very quickly [elixir is becomming a huge thing]
- https://medium.com/@kenmazaika/why-im-betting-on-elixir-7c8f847b58
- https://medium.com/@j_mcnally/the-peep-stack-b7ba5afdd055

Elixir - is it worth it?

Let's face it. CRUD apps are a commodity these days. The next: *AirBnB for Renting Ketchup* probably isn't going to survive.

The people who will win are going to be the ones who embrace changes in technology. The fact that WebSockets, processes, and concurrency in Phoenix and Elixir are cheap, without sacrificing developer happiness is an absolute game-changer.

Elixir links

- http://learnxinyminutes.com/docs/elixir/
- http://elixirdose.com/
- https://teamgaslight.com/blog/the-best-resources-for-learning-elixir
- http://elixirsips.com/
- http://learnyousomeerlang.com/content
- https://github.com/h4cc/awesome-elixir

More Elixir links

- https://speakerdeck.com/michalslaski/introduction-to-erlang
- https://github.com/elixir-lang/elixir/wiki/Talks
- “Individualised Content at Web-scale” (slides / video)
- http://www.theerlangelist.com/2014/01/why-elixir.html

Thanks

Authors:

- Jacek Tomaszewski (jtom.me)
- Karol Wojtaszek (appunite.com)