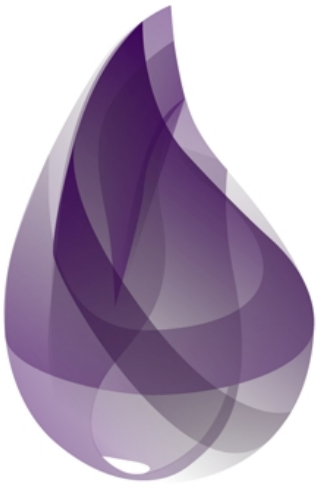
EXPRESSIVIDADE E CONCORRÊNCIA COM ELIXIR



elixir

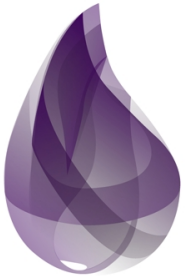
AGENDA

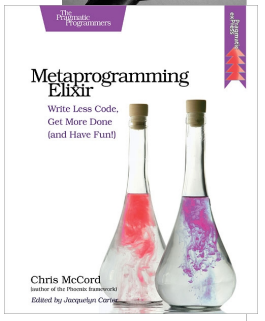
- INTRODUÇÃO
- A LINGUAGEM
- LINHA DO TEMPO
- CONCORRÊNCIA
- ELIXIR x RUBY
- CONCLUSÃO
- BIBLIOGRAFIA

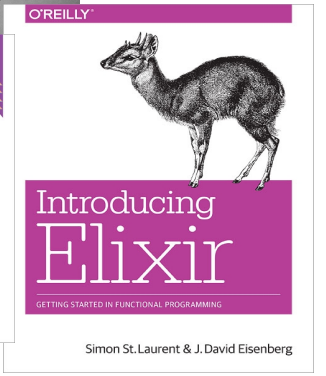
INTRODUÇÃO

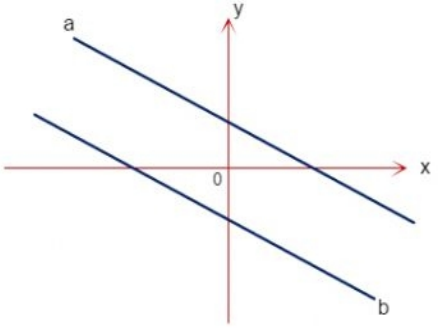

ORBO


ORDEM E PROGRESSO




The Pragmatic Programmers
Metaprogramming Elixir
Write Less Code, Get More Done (and Have Fun!)
Chris McCord

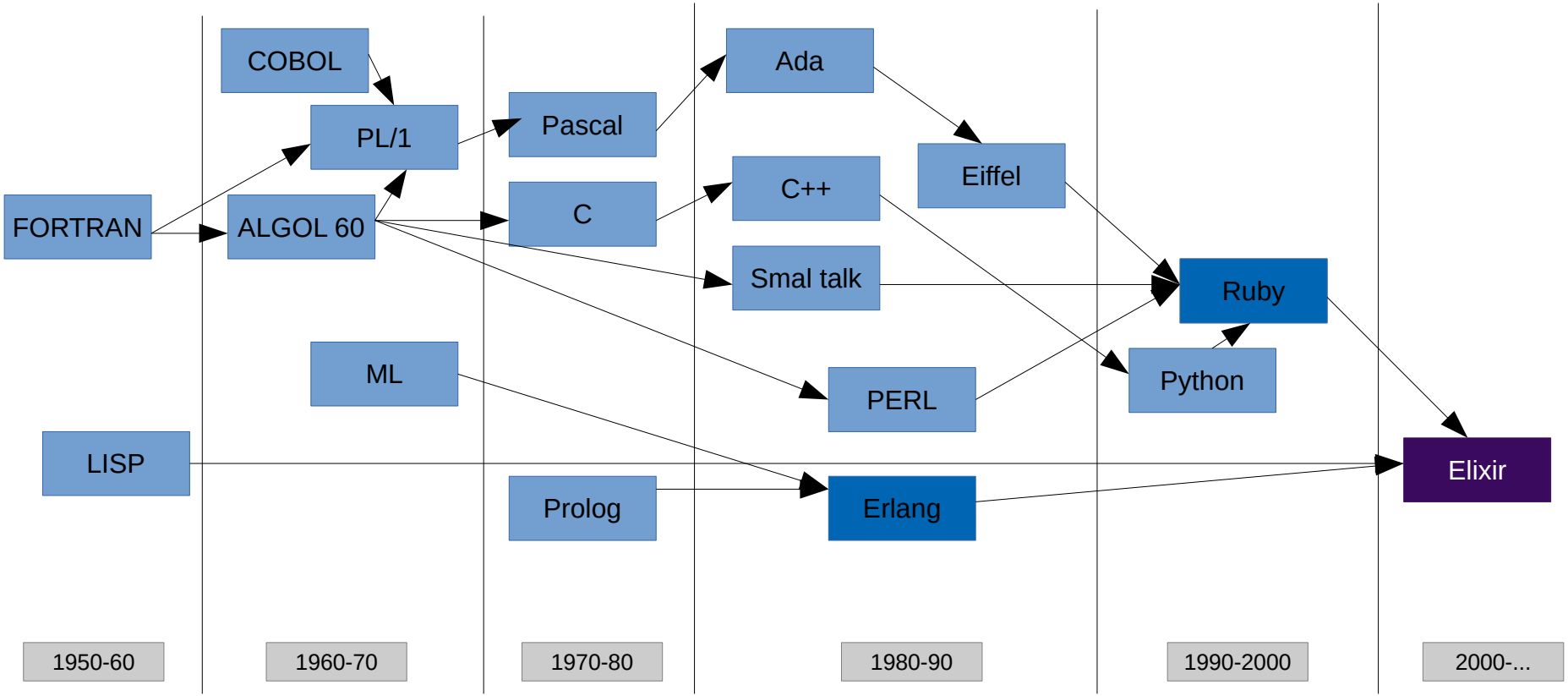

O'REILLY
Introducing Elixir
GETTING STARTED IN FUNCTIONAL PROGRAMMING
Simon St.Laurent & J. David Eisenberg


a
y
x
0
b

A LINGUAGEM

- Linguagem de programação funcional, concorrente e de propósito geral
- Executada e compilada para bytecode dentro da máquina virtual Erlang (BEAM)
- Funções de Erlang podem ser chamadas de Elixir no tempo de execução.
- A Erlang foi desenvolvida em 1986, pela Ericsson.
- Suporta metaprogramação com macros e polimorfismo através de protocolos.
- Usada desenvolvimento web.
- Construção de sistemas embarcados.
- Suporte para documentação via docstrings tipo Python na linguagem de formatação
- Ênfase na recursão e funções de ordem superior.
- Suporte a Unicode e cadeias UTF-8
- Sua aplicação pode responder a requisições independente de quantos clientes estão conectados.
- Nunca quebra.
- Código fácil de crescer e manter, implicando em baixo custo.

LINHA DO TEMPO



A LINGUAGEM

- inteiros
- reais (float)
- booleanos
- atomos
- cadeias (strings)
- listas
- tuplas

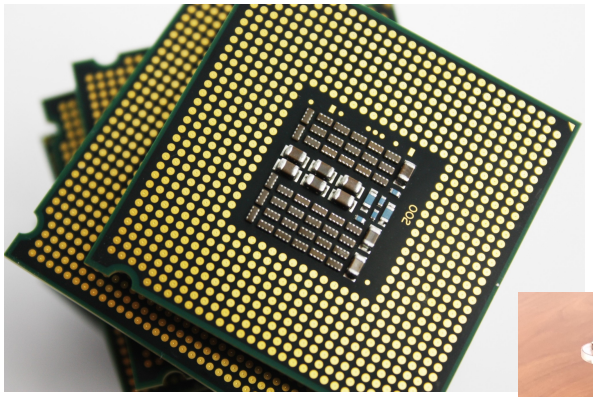
```
iex(6)> IO.puts "Hello world!"
Hello world!
:ok
iex(7)> IO.puts "Hello {#world}!"
Hello {#world}!
:ok
```

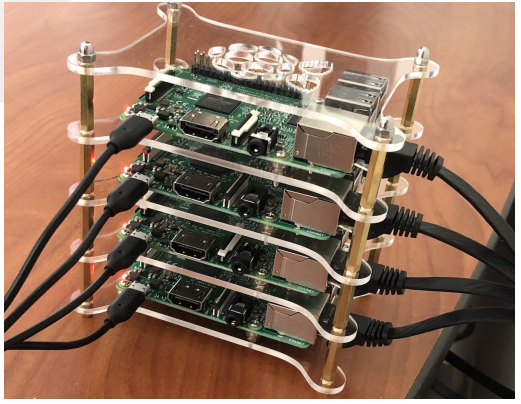
```
iex(8)> is_binary("Olá!")
true
iex(9)> byte_size("Olá!")
5
```

```
iex(10)> 3+3
6
iex(11)> add = fn a, b -> a + b end
#Function<12.99386804/2 in :erl_eval.expr/5>
iex(12)> add.(3,3)
6
```

```
iex(13)> [1, 2, true, 3]
[1, 2, true, 3]
iex(14)> length([1, 2, true, 3])
4
iex(15)> [1,2,3]++[4,5,6]
[1, 2, 3, 4, 5, 6]
iex(16)> [1, true, 2, false, 3, true]--[true,false]
[1, 2, 3, true]
```

CONCORRÊNCIA

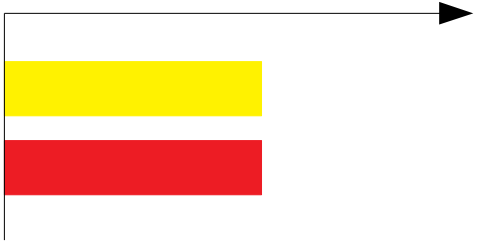




- Processadores multi-core

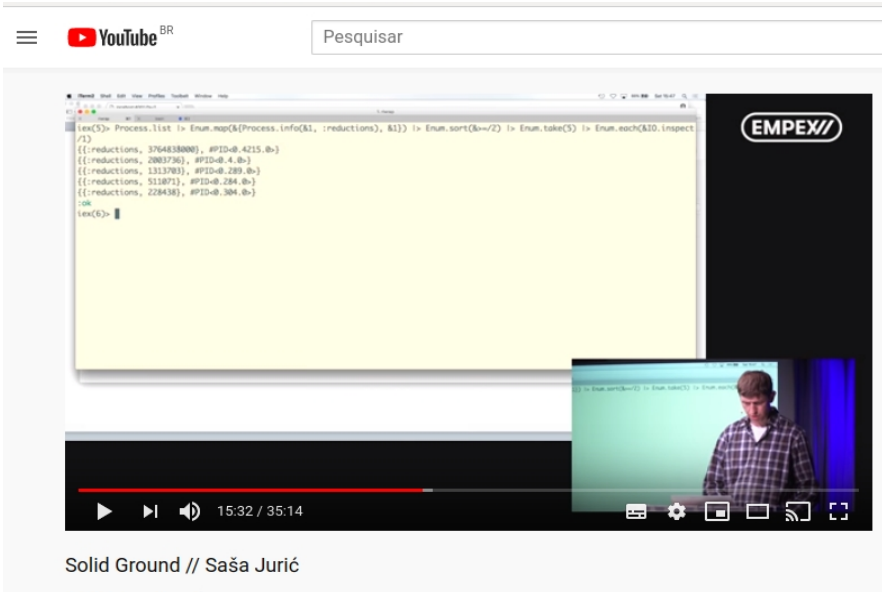
- **Paralelismo:** Acontece quando duas ou mais tarefas são livres para serem executadas, literalmente, ao mesmo tempo.

-**Concorrência:** Quando duas ou mais tarefas podem começar a ser executadas e terminar em espaços de tempo que se sobrepõem, não significando que elas precisam estar em execução necessariamente no mesmo instante.



CONCORRÊNCIA

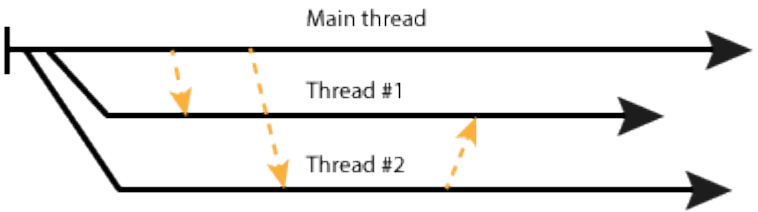
- Demonstração de um caso prático do Elixir, atuando em um caso real, incluindo a correção de um "bug".



https://www.youtube.com/watch?v=pO4_Wlq8Jel

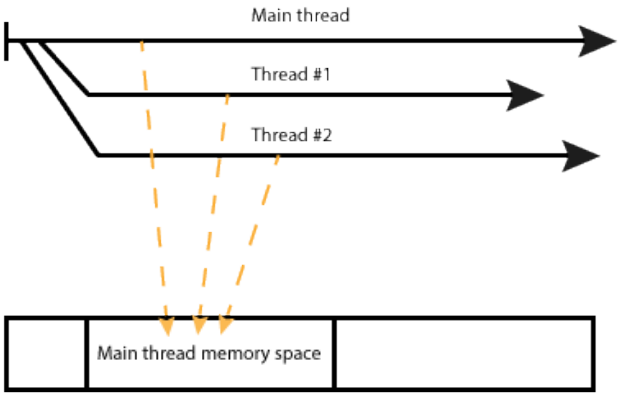
ELIXIR X RUBY

Elixir



Usa Plataforma Concorrentel
Inspirada na linguagem Ruby.
"Share nothing"

Ruby



Concorrência não era real.
Não tem threads nativas.
Global Interpreter Lock (GIL).
Não aproveita tanto vantagens dos multi-core.
Actor Model
A comunicação "Queue Class", compartilhada.

ELIXIR X RUBY

Elixir

```elixir
defmodule NumberPrinter do
  def print_numbers(thread_number) do
    Enum.each 1..5, fn(j) ->
      IO.puts "Thread: #{thread_number}, number: #{j}"
      :timer.sleep(Enum.random(0..500))
    end
  end
end

Enum.each 1..5, fn(thread_number) ->
  spawn(NumberPrinter, :print_numbers, [thread_number])
end
```

```
...(3)> end
Thread: 3, number: 1
Thread: 1, number: 1
Thread: 2, number: 1
Thread: 4, number: 1
Thread: 5, number: 1
:ok
```

Ruby

```ruby
def print_numbers(thread_number)
  (0..5).each do |j|
    p "Thread: #{thread_number}, number: #{j}"
    sleep(Random.rand)
  end
end

(0..5).each do |i|
  Thread.new { print_numbers(i) }
end
```

```
(base) bruno@bruno-Latit
"Thread: 2, number: 0"
"Thread: 1, number: 0"
"Thread: 0, number: 0"
"Thread: 4, number: 0"
"Thread: 5, number: 0"
"Thread: 3, number: 0"
```

CONCLUSÃO

- Nasceu projetada para lidar com concorrência.
- Máximo potencial do hardware.
- Alta capacidade de processamento paralelo.
- Pouco susetível à falhas, alta disponibilidade
- Escalabilidade
- Expressividade no código
- Ganhando mercado, baixo custo e fácil manutenção.

Bibliografia

-https://pt.wikipedia.org/wiki/Elixir_(linguagem_de_programa%C3%A7%C3%A3o)

- https://elixir-lang.org/getting-started/processes.html

- https://medium.com/full-stack-tips/a-gentle-introduction-to-elixir-1da9261bae72

- https://videosdeti.com.br/por-que-voce-tambem-deve-aprender-elixir/

- https://elixirschool.com/pt/lessons/basics/basics/

- http://zonov.me/go-for-rubyists-part-8-concurrency-ruby-go-elixir/

- DAVI, Tiago. Elixir: Do zero à concorrência.São Paulo:Editora Caso do Código.2017.

- https://lucassimon.com.br/2018/06/um-caminho-para-aprender-elixir/

- http://www.tiagobarreto.com/comecando-com-elixir-e-phoenix/

- http:// jlouisramblings.blogspot.com/2013/01/how-erlang-does-scheduling.html

- https://diogommartins.wordpress.com/2017/04/07/concorrencia-e-paralelismo-threads-multiplos-processos-e-asyncio-parte-1/

- https://en.wikipedia.org/wiki/Elixir_(programming_language)

- http://zonov.me/go-for-rubyists-part-8-concurrency-ruby-go-elixir/

OBRIGADO !