Pagantis

acm

Una gota de productividad en tu trabajo

Phoenix Framework™

@kamiyuzu

@hiimkamiyuzu

Sobre mi

- Alberto Revuelta Arribas
- Ingeniería informática en la universidad politécnica de Madrid.
- Trabajo actual: Desarrollador de software (Elixir) en Pagantis.

Índice

Historia de elixir

- Ruby (1995)
- José Valim (creador de elixir)

[Concurrency in Ruby: In search of inspiration]

Historia de elixir (2011)

"José – Before writing Elixir, I already had a good knowledge of the Erlang language and the Erlang VM. At the time, I was unhappy with the tools available to solve the concurrency issues in the Ruby ecosystem and the fantastic Seven Languages in Seven Weeks (1) book helped me lay out the options out there. After reading the book, it solidified my opinion that the Erlang VM is one of the best environments to build and deploy robust concurrent software (my goals at the time)." - Talking about Elixir and the Erlang VM with José Valim

Elixir: The Documentary

Historia de elixir

- **Erlang (1986)**
- **EEF (Erlang Ecosystem Foundation)**
    - Laura M. Castro (profesora e investigadora en la universidade da Coruña)

Erlang Ecosystem Foundation

Elixir

- Basic types:

```
iex> 1              # integer
iex> 0x1F           # integer
iex> 1.0            # float
iex> true           # boolean
iex> :atom          # atom / symbol
iex> "elixir"       # string
iex> [1, 2, 3]      # list
iex> {1, 2, 3}      # tuple
```

Elixir

- Data structures:
- Keyword list:

```
iex> list = [{:a, 1}, {:b, 2}]
[a: 1, b: 2]
iex> list == [a: 1, b: 2]
true
```

- Maps:

```
iex> map = %{:a => 1, 2 => :b}
%{2 => :b, :a => 1}
iex> map[:a]
1
iex> map[2]
:b
iex> map[:c]
nil
```

Elixir

- Pattern matching:

```
iex> x = 1
1
iex> 1 = x
1
iex> 2 = x
** (MatchError) no match of right hand side  value: 1
iex> 1 = unknown
** (CompileError) iex:1: undefined function unknown/0
```

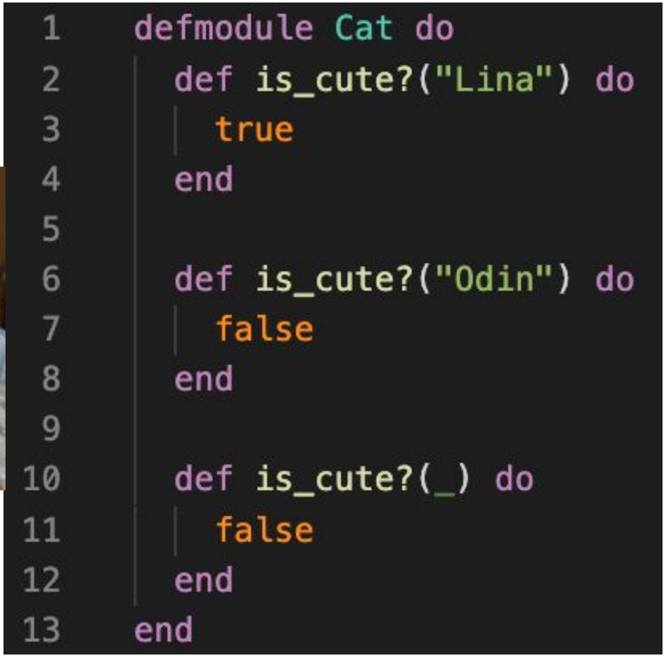
Elixir

- Pattern matching:

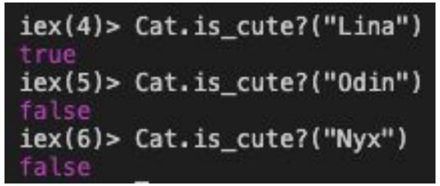
```
iex> {a, b, c} = {:hello, "world", 42}

{:hello, "world", 42}

iex> a

:hello

iex> b

"world"
```

Elixir

- Pattern matching methods:



```
defmodule Cat do
  def is_cute?("Lina") do
    true
  end

  def is_cute?("Odin") do
    false
  end

  def is_cute?(_) do
    false
  end
end
```



```
iex(4)> Cat.is_cute?("Lina")
true
iex(5)> Cat.is_cute?("Odin")
false
iex(6)> Cat.is_cute?("Nyx")
false
```

Elixir

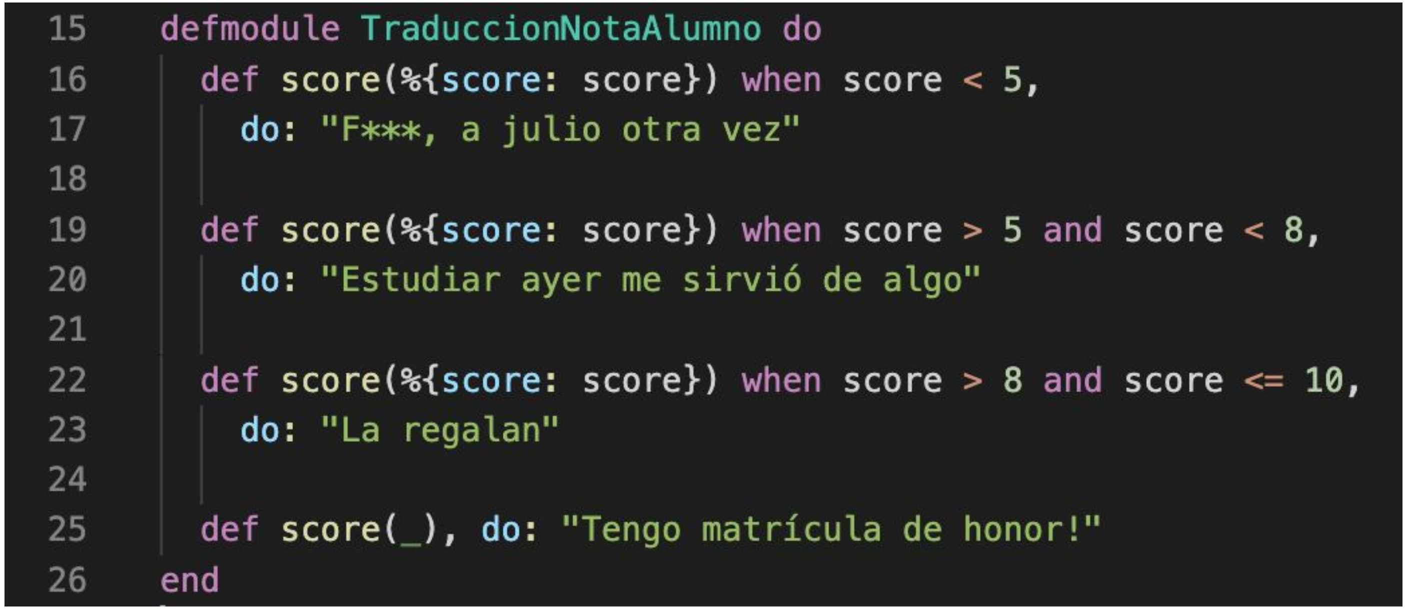
- **Guards:**

You can find the built-in list of guards in the Kernel module. Here is an overview:

- comparison operators (==, !=, ===, !==, >, >=, <, <=)
- strictly boolean operators (and, or, not). Note &&, ||, and ! sibling operators are not allowed as they're not *strictly* boolean - meaning they don't require arguments to be booleans
- arithmetic unary and binary operators (+, -, +, -, *, /)
- in and not in operators (as long as the right-hand side is a list or a range)
- "type-check" functions (is_list/1, is_number/1, etc.)
- functions that work on built-in datatypes (abs/1, map_size/1, etc.)

Elixir

- Guards:



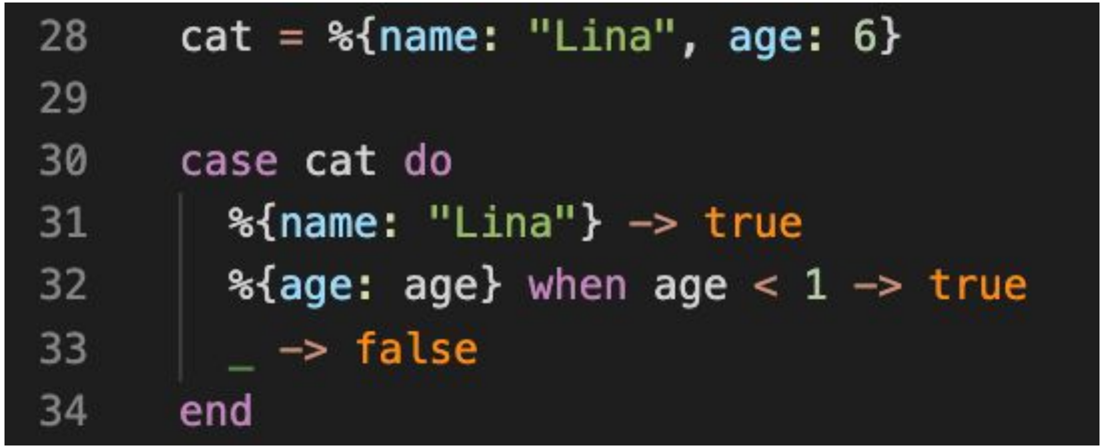
```elixir
defmodule TraduccionNotaAlumno do
  def score(%{score: score}) when score < 5,
    do: "F***, a julio otra vez"

  def score(%{score: score}) when score > 5 and score < 8,
    do: "Estudiar ayer me sirvió de algo"

  def score(%{score: score}) when score > 8 and score <= 10,
    do: "La regalan"

  def score(_), do: "Tengo matrícula de honor!"
end
```

Elixir

- Conditionals:
    - Case
    - Cond
    - If/else

Elixir

- Case:



```
cat = %{name: "Lina", age: 6}

case cat do
  %{name: "Lina"} -> true
  %{age: age} when age < 1 -> true
  _ -> false
end
```

Elixir

- **Cond:**

```
iex> cond do
...>   2 + 2 == 5 ->
...>     "This will not be true"
...>   2 * 2 == 3 ->
...>     "Nor this"
...>   1 + 1 == 2 ->
...>     "But this will"
...> end
```

```
iex> "But this will"
```

Pagantis

Elixir

- If/else:

```
iex> if nil do
...>    "This won't be seen"
...> else
...>    "This will"
...> end
iex> "This will"
```

- Es una macro
- No está disponible en Erlang

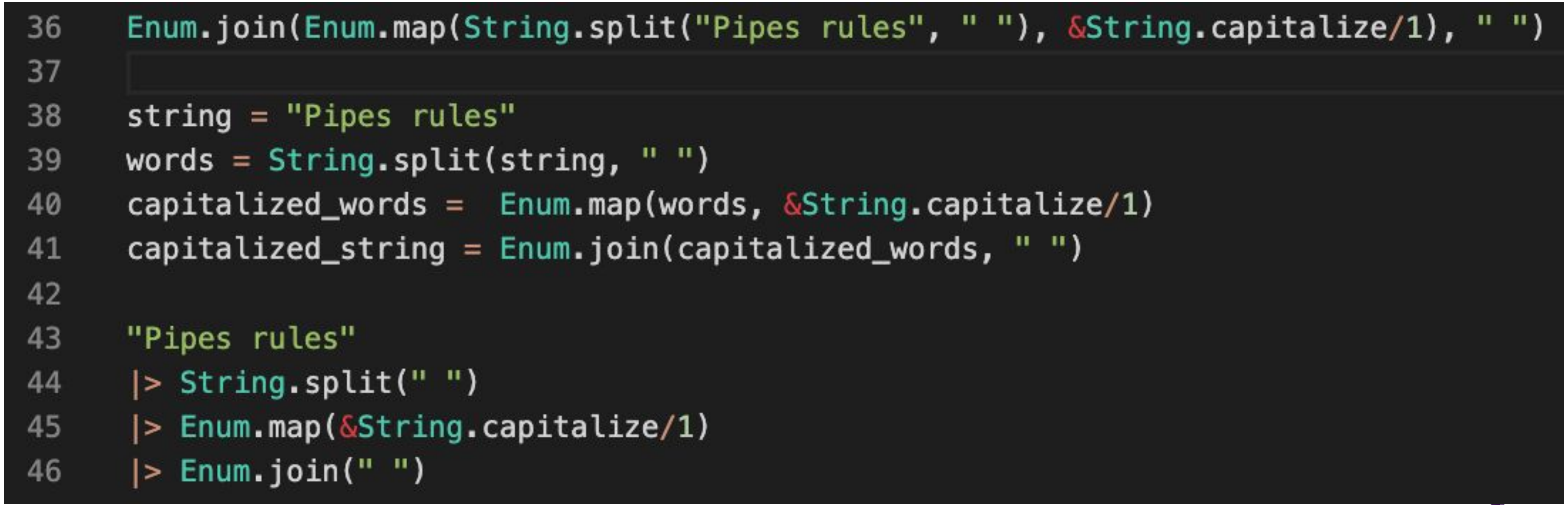
Pagantis

Elixir

- **Azúcar sintáctico:**
    - Pipe operator
    - With statement

Elixir

- Pipe operator:



```
Enum.join(Enum.map(String.split("Pipes rules", " "), &String.capitalize/1), " ")

string = "Pipes rules"
words = String.split(string, " ")
capitalized_words =  Enum.map(words, &String.capitalize/1)
capitalized_string = Enum.join(capitalized_words, " ")

"Pipes rules"
|> String.split(" ")
|> Enum.map(&String.capitalize/1)
|> Enum.join(" ")
```

Elixir

- With statement: (macro)

```
iex> user = %{first: "Sean", last: "Callan"}
%{first: "Sean", last: "Callan"}
iex> with {:ok, first} <- Map.fetch(user, :first),
...>      {:ok, last} <- Map.fetch(user, :last),
...>      do: last <> ", " <> first
"Callan, Sean"
```

Elixir

- **With statement:**

```
iex> user = %{first: "doomspork"}
%{first: "doomspork"}
iex> with {:ok, first} <- Map.fetch(user, :first),
...>      {:ok, last} <- Map.fetch(user, :last),
...>      do: last <> ", " <> first
:error
```

Buenas prácticas

- Buenas prácticas de código: credo
- Herramienta de análisis sintáctico: dialyzer
- Formateo del código: mix format

TDD

- Refactoring: improving the design of existing code (2)
- Test-driven development (3)

DDD

- Lenguaje ubicuo
- Crafted design - Sandro Mancuso
- Domain-driven design distilled (4)

Demo

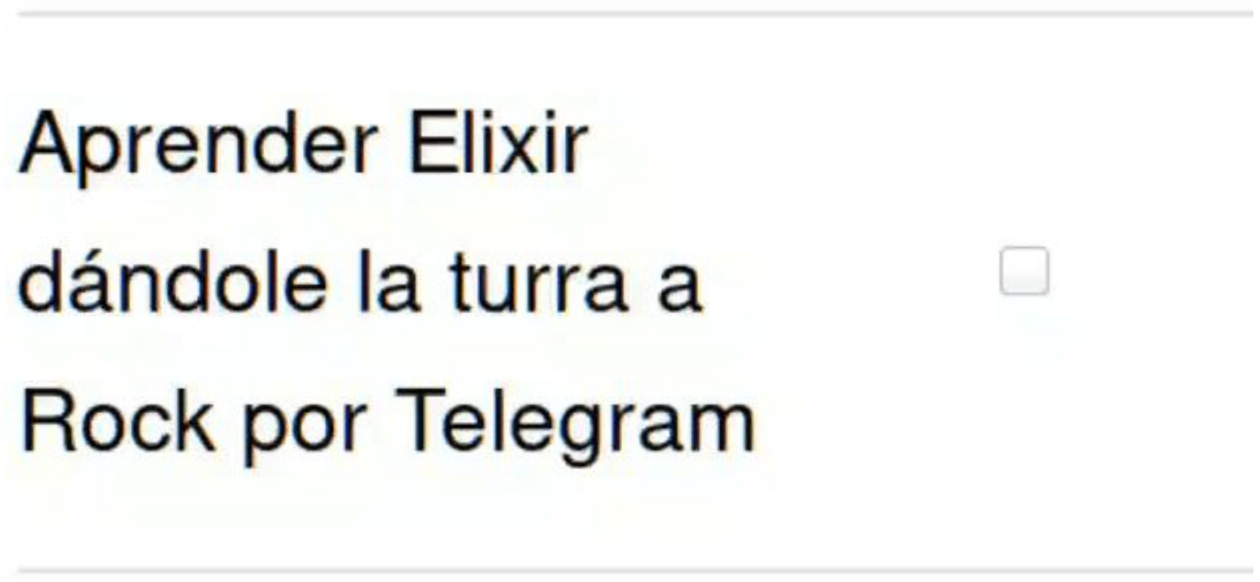
Recursos de Elixir

- Elixir wizards (podcast)
- Elixir reddit (reddit url link)
- Code sync (youtube channel)
- Dashbit blog (Blog posts from José Valim)
- Elixir in action (5) (Elixir book)
- Elixir status (newsletter)
- Elixir radar (newsletter)
- Advent of code (José Valim twitch)
- Design principles: elixir S.O.L.I.D. (youtube video)

- Tail recursive Disney mashup song (!!Con 2019)
- The little Elixir OTP guidebook (6) (Openlibra telegram link)

Bibliografía

1) Tate, B., 2010. *Seven Languages In Seven Weeks*. Raleigh, N.C.: Pragmatic Bookshelf.
2) Fowler, M. and Beck, K., 2018. *Refactoring: Improving The Design Of Existing Code*. 2nd ed. Boston [etc.]: Addison-Wesley.
3) Beck, K., 2003. *Test-Driven Development*. Boston: Addison-Wesley.
4) Vernon, V., 2016. *Domain-Driven Design Distilled*. Pearson Technology Group Canada.
5) Jurić, S., 2015. *Elixir In Action*. Shelter Island, NY: Manning.
6) Wei, B., 2016. *Little Elixir & OTP Guidebook*. Manning Publications.



Aprender Elixir dándole la turra a Rock por Telegram ☐

@rockneurotiko