Programação Funcional com Elixir

Pattern Matching

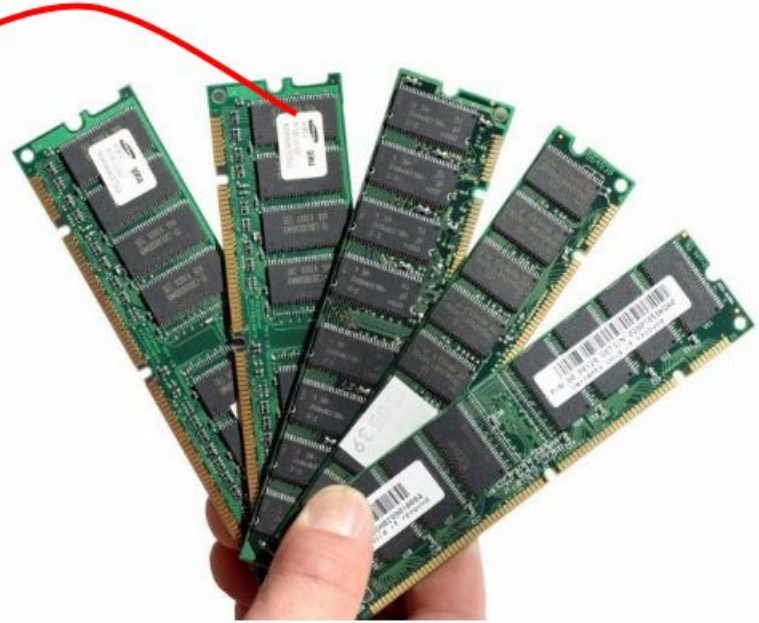
Pattern Matching

- A primeira coisa que precisamos aprender sobre Pattern Matching é que o "=" não é um operador de atribuição no Elixir. Veja:
  - `iex(1)> n1 = 1`
  - `1`
  - `iex(2)> 1 = n1`
  - `1`
  - `iex(3)> 2 = n1`
  - `** (MatchError) no match of right hand side value: 1`

Pattern Matching

- Isso ocorre pois **n1** não "**atribui**" e sim "**aponta para**" **1**

FFF1A: N1 FFF1D	FFF1B	FFF1C	FFF1D 1	FFF1E
FFF1F	FFF2A	FFF3A	FFF4A	FFF5A
FFF6A	FFF6A	FFF8A	FFF8B	FFF4B

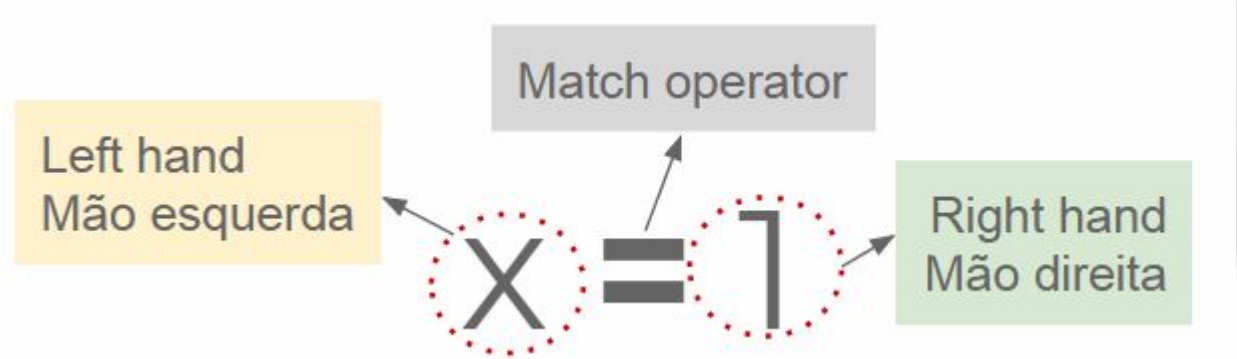


Pattern Matching

- Observe que a operação inversa também foi válida:
  - `iex(2)> 1 = n1`
  - `1`
- Neste caso não é válida pois o “=” aqui na verdade é o **Match Operator**.

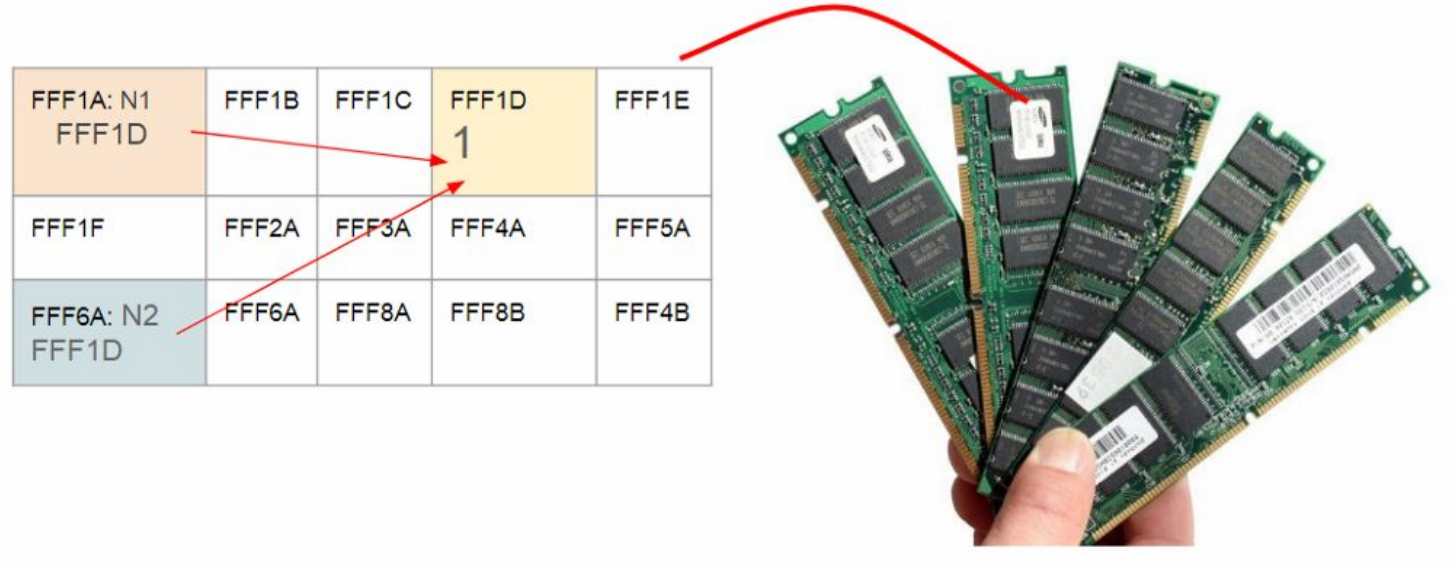
Pattern Matching

- O Match Operator (=) é um operador binário, ou seja, ele precisa ter dois elementos para serem avaliados (um de cada lado).



Pattern Matching

- Vejamos agora o que acontece...
  - `iex(4)> n2 = n1`
  - `1`



Pattern Matching

- O Match Operator só "atribui" variáveis do lado esquerdo do operador match.

Pattern Matching

- Agora, que já entendemos o Match Operator, vamos brincar com o Pattern Matching, que tem o mesmo princípio mas pode ser aplicado a estruturas mais complexas.

Pattern Matching

```
iex(1)> {a, b, c} = {:jackson, "pires", 123}
{:jackson, "pires", 123}

iex(2)> a
:jackson

iex(3)> b
"pires"

iex(4)> c
123
```

Pattern Matching

- Perceba no exemplo anterior, do lado esquerdo temos uma tupla constituída apenas de variáveis, e do lado direito uma tupla com alguns valores.
- O Elixir verifica se as estruturas podem ser correspondidas e em caso positivo faz as atribuições.
- Caso as estruturas não sejam equivalentes, um erro ocorrerá.

Pattern Matching

```
iex> {a, b, c} = {:jackson, "pires"}
** (MatchError) no match of right hand side value: {:jackson,
"pires"}

iex> {a, b, c} = [:jackson, "pires", 123]
** (MatchError) no match of right hand side value: [:jackson,
"pires", 123]
```

Pattern Matching

- Outra coisa interessante que podemos usar com Pattern Matching é a estrutura de cabeça e cauda para listas.

```
iex(1)> [cabeca | cauda] = [1, 2, 3]
[1, 2, 3]

iex(2)> cabeca
1

iex(3)> cauda
[2, 3]
```

Pattern Matching

- Assim, aprendemos mais um dos princípios do paradigma funcional.
  - First-Class functions ✔
  - Pure functions
  - Immutable variables ✔
  - Recursion
  - Nonstrict evaluation
  - Statements ✔
  - Pattern matching ✔