



Programação Funcional com Elixir

The background features a series of overlapping, semi-transparent gray shapes that resemble stylized, flowing liquid or smoke. These shapes are concentrated on the left side of the frame, creating a sense of movement and depth. The right side of the image is a plain, light gray.

Pipe Operator

Pipe Operator

- Para entender o Pipe Operator, vamos desconstruir esse exemplo...
 - `IO.puts(String.length("Olá"))`

Pipe Operator

- A primeira coisa que podemos fazer para melhorar a legibilidade e entendimento é separá-la em dois passos:
 - `str_len = String.length("Olá")`
 - `IO.puts(str_len)`

Pipe Operator

- Um próximo refatoramento que podemos fazer é entender e usar o pipe operator `|>`
 - `str_len = String.length("Olá")`
 - `IO.puts(str_len)`
 - **`String.length("Olá") |> IO.puts`**

Pipe Operator

“O pipe operator permite que o resultado da expressão anterior seja o valor para o primeiro parâmetro da expressão seguinte.”

Pipe Operator

- Sendo assim, podemos desconstruir ainda mais...
 - `str_len = String.length("Olá")`
 - `IO.puts(str_len)`
 - `String.length("Olá") |> IO.puts`
 - `"Olá!" |> String.length |> IO.puts`

Pipe Operator

- Por fim, podemos organizar de uma forma mais legível...
 - `"Olá!"`
 - `|> String.length`
 - `|> IO.puts`