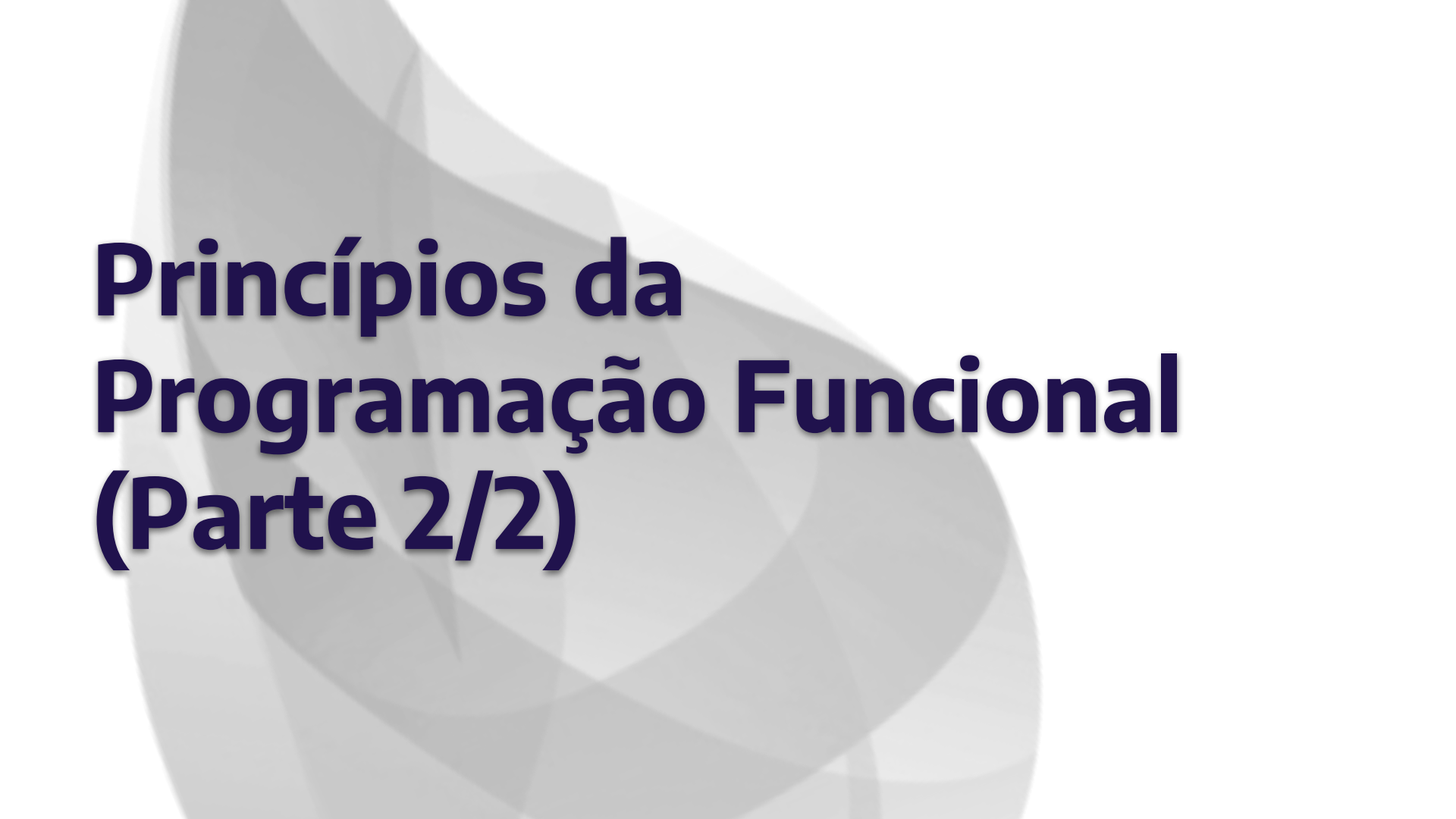




# **Programação Funcional com Elixir**



# **Princípios da Programação Funcional (Parte 2/2)**

# Princípios da Programação Funcional

- ~~First Class functions~~
- ~~Pure functions~~
- ~~Immutable variables~~
- ~~Recursion~~
- Nonstrict evaluation
- Statements
- Pattern matching

The background features a series of overlapping, semi-transparent gray shapes that resemble stylized leaves or petals, creating a layered, organic effect. The shapes are primarily concentrated on the left side of the frame, with some extending towards the center.

# **Nonstrict Evaluation**

# Nonstrict Evaluation

- **Avaliações não restritas** nos permitem ter variáveis que ainda não foram computadas.
- Por outro lado, Strict Evaluations (Avaliações Rigorosas) atribuem uma variável assim que ela é definida, que é o que estamos acostumados.
- Nonstrict significa que podemos ter uma variável que não seja atribuída (computada) até a primeira vez em que é referenciada.

# Nonstrict Evaluation

- Nesse ponto podemos lembrar do Haskell que sempre levanta a bandeira do “Lazy Evaluation” (Avaliação Preguiçosa), “Delayed Evaluation” (Avaliação Tardia), ou ainda Call-by-name (Chame pelo nome) que a grosso modo são mesma coisa.
- Perceba que isso é inerente à linguagem de programação, não dependendo do programador.

# Nonstrict Evaluation

## Lazy Evaluation

Lazy evaluation (or call-by-name) is an evaluation strategy which delays the evaluation of an expression until its value is needed



The background of the slide 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 slide is a plain, light gray.

# Statements



# Statements

- As **declarações** são pedaços de código “avaliáveis” que possuem um valor de retorno.
- Pense em instruções "if" que tenham algum valor de retorno.
- Cada linha de código deve ser considerada uma declaração.

```
puts("O valor de retorno é #{x=1}")  
O valor de retorno é 1
```

# Statements

- Ou seja, a programação funcional introduz a ideia de que cada linha de código deve retornar um valor.
- Então, se estamos aptos a fazer mais uso de statements nós podemos reduzir o número de variáveis e se reduzimos a quantidade de variáveis, reduziremos também a necessidade de “mudá-las” e isso aumenta a habilidade de executar processos concorrentes e torna-se mais funcional.

The background of the slide features a series of overlapping, semi-transparent gray shapes that resemble stylized, flowing petals or abstract organic forms. These shapes are layered, creating a sense of depth and movement, primarily concentrated on the left side of the frame.

# **Pattern Matching**

# Pattern Matching

- A "**correspondência de padrões**" não aparece realmente na matemática, mas ajuda a programação funcional a diminuir a necessidade de variáveis específicas.
- Ou ainda, Pattern matching nos permite procurar padrões simples em valores, estruturas de dados, e até funções.

# Pattern Matching

- No código, geralmente encapsulamos um grupo de variáveis juntas dentro de um objeto. A correspondência de padrões nos permite obter uma melhor verificação de tipos e extrair elementos de um objeto, criando instruções mais simples e concisas com menos necessidade de definições de variáveis.

# Pattern Matching



# Pattern Matching

Text : A A B A A C A A D A A B A A B A

Pattern : A A B A

A A B A

A A B A

A A B A A C A A D A A B A A B A  
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15  
A A B A

Pattern Found at 1, 9 and 12

# Princípios da Programação Funcional

- First-Class functions
- Pure functions
- Immutable variables
- Recursion
- Nonstrict evaluation
- Statements
- Pattern matching