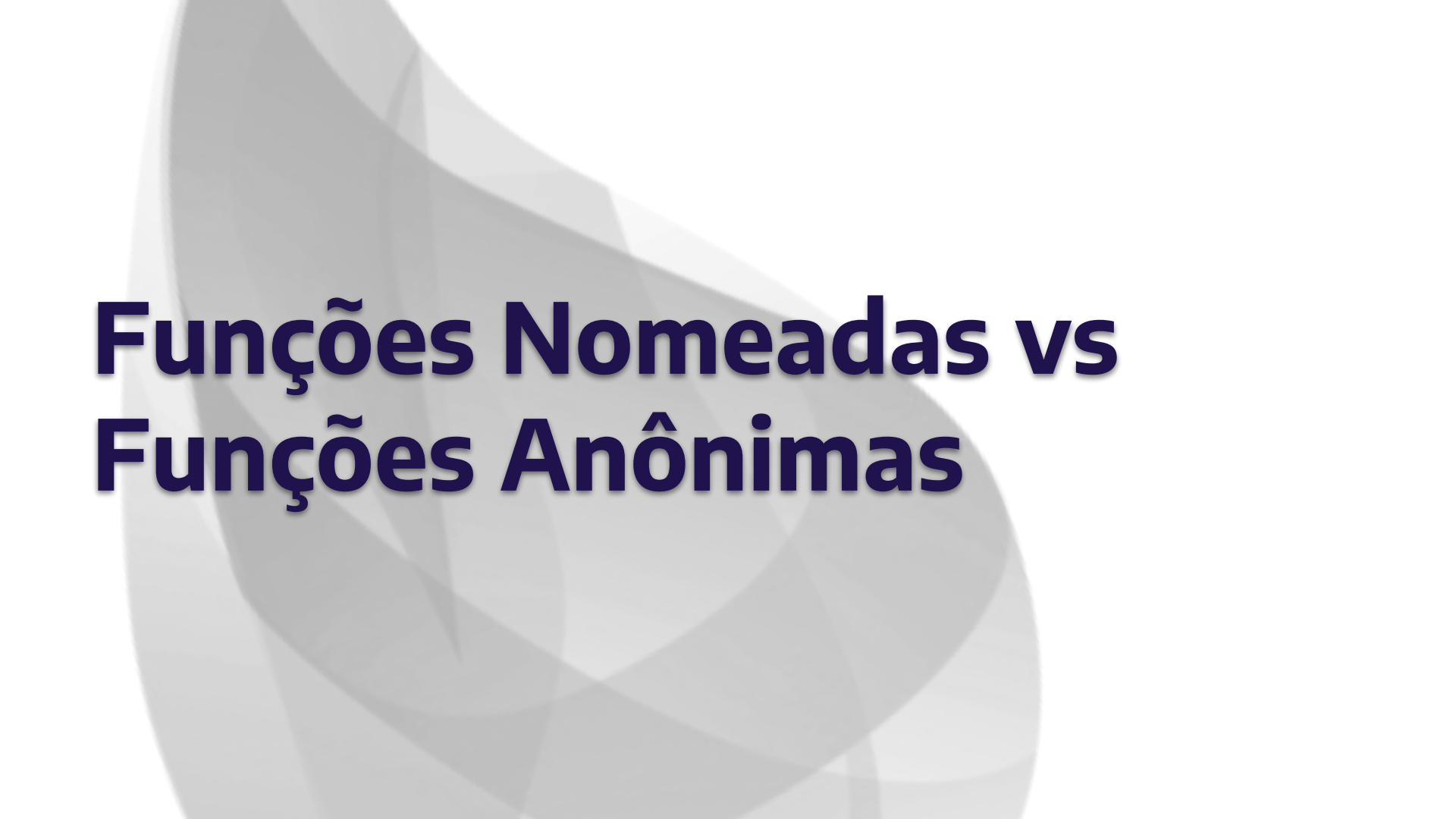




# **Programação Funcional com Elixir**



# **Funções Nomeadas vs Funções Anônimas**

# Funções Nomeadas vs Funções Anônimas

- Até agora usamos “funções nomeadas”, que basicamente são funções possuem um nome.
- As funções anônimas são funções definidas sem um nome atrelado, mas que podem ser atribuídas (bind) a uma variável. Veja

- `sum = fn (a, b) -> a + b end`

# Funções Nomeadas vs Funções Anônimas

- Analisando uma função anônima, temos:

○ `sum = fn (a, b) -> a + b end`

Palavra-chave

Parâmetros

Corpo da  
função

Finalizador

# Funções Nomeadas vs Funções Anônimas

- Para usar uma função anônima devemos usar o ponto (.) no momento de passar os argumentos. Veja:
  - `sum = fn (a, b) -> a + b end`
  - `sum.(2, 3)`

# Funções Nomeadas vs Funções Anônimas

- Para múltiplas instruções no corpo da função use “;” ou múltiplas linhas:

- `printed_sum = fn (a, b) -> c = a + b;  
 IO.puts(">>#{c}<<") end`
- 
- `printed_sum = fn (a, b) ->`
- `c = a + b`
- `IO.puts(">>#{c}<<")`
- `end`

# Funções Nomeadas vs Funções Anônimas

- Podemos também remover os parênteses. Veja:
  - `hello = fn name -> "Hello, #{name}!" end`
  - `hello.("Ana")`

# Funções Nomeadas vs Funções Anônimas

- Podemos também criar funções anônimas sem parâmetros:

- `one_plus_one = fn -> 1 + 1 end`
- `one_plus_one.()`