Getting Started with Elixir

João Gonçalves

Video 4.2

Pattern Matching in Functions

Getting Started with Elixir

João Gonçalves

Video 4.2

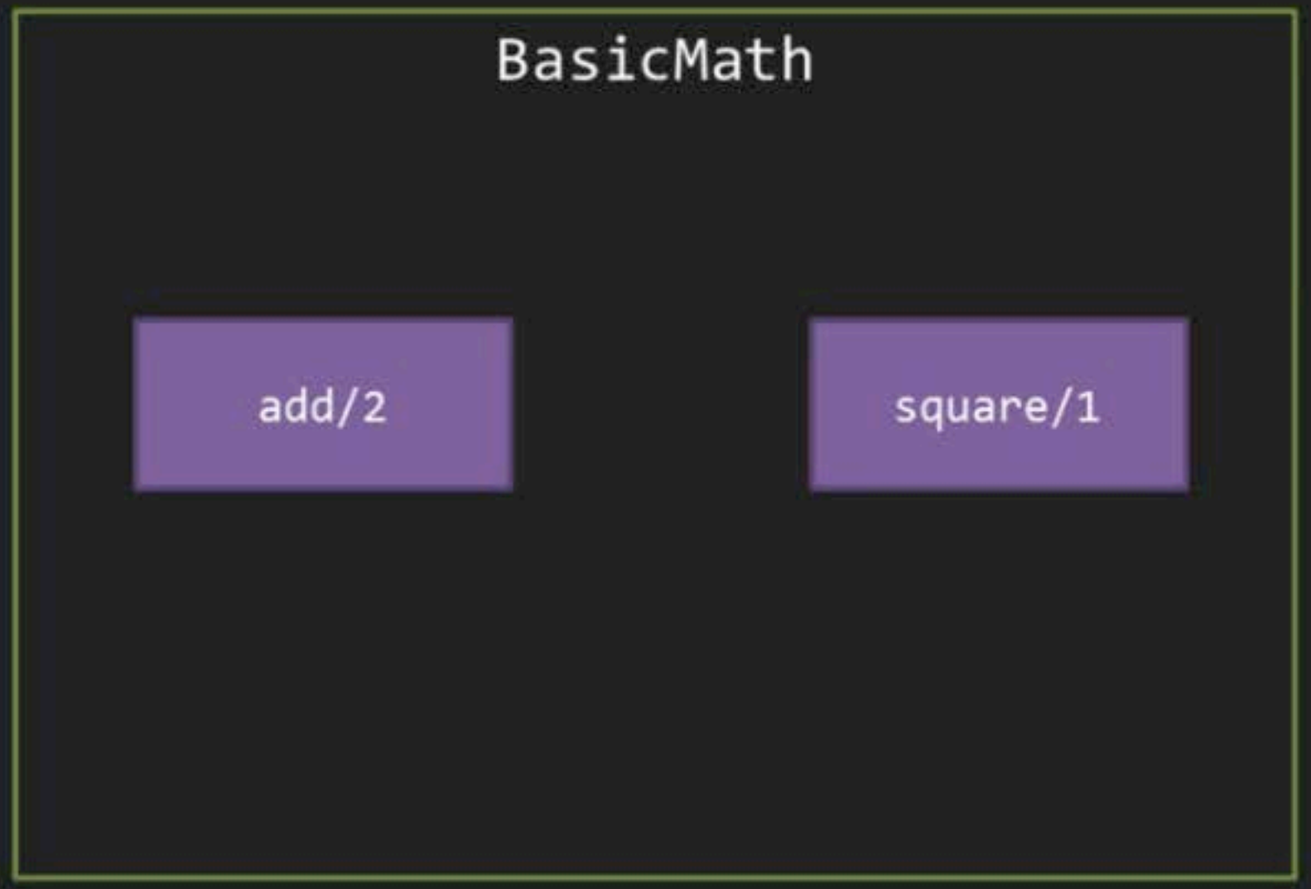
Pattern Matching in Functions

Packt

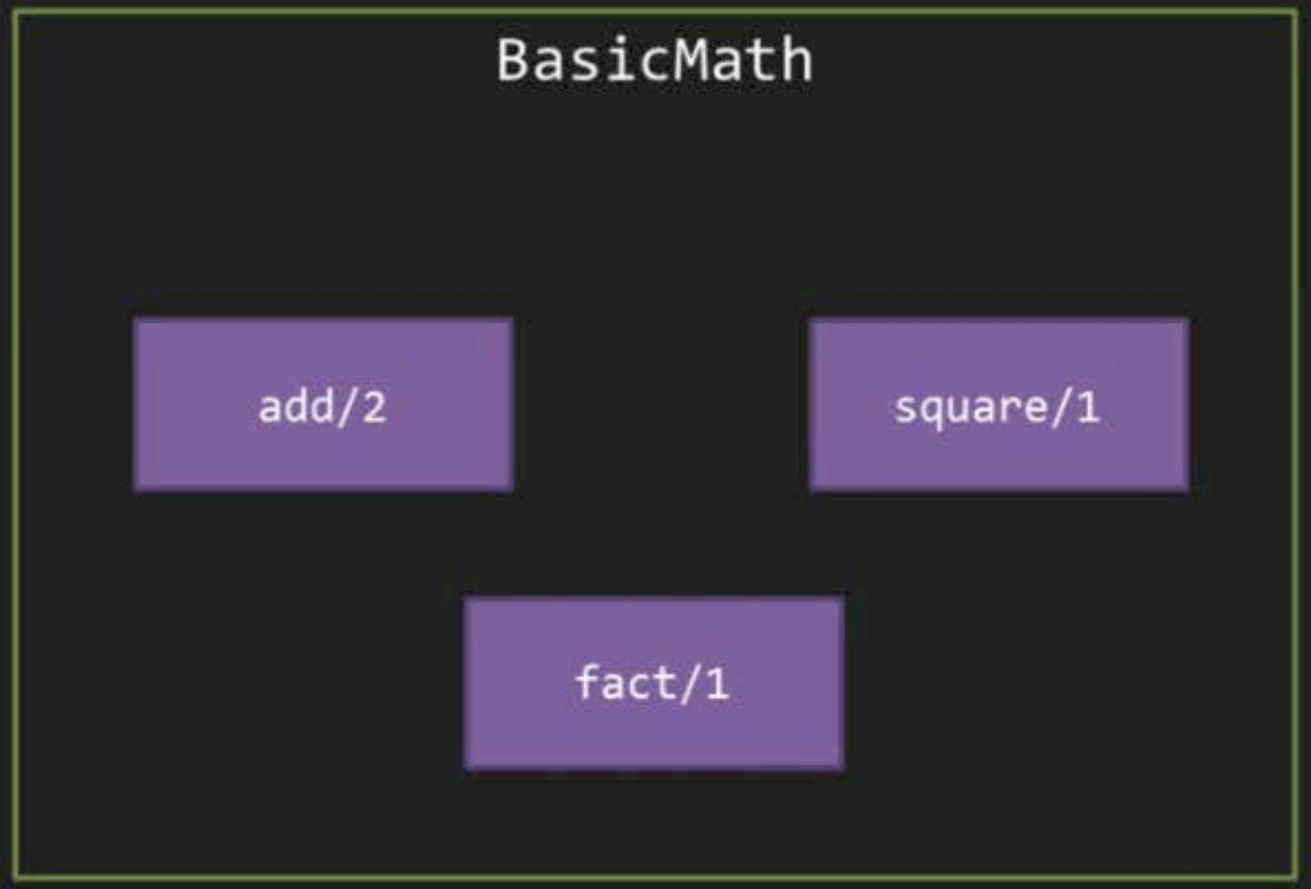
In this video, we are going to take a look at...

- How to leverage pattern matching in function calls
- Using function guards

Extending BasicMath



Extending BasicMath



The Factorial Function

$$n! = \prod_{k=1}^{n} k$$

The Factorial Function

$$n! = f(n) = \begin{cases} n \times f(n-1) & if\ n > 0 \\ 1\ if\ n \leq 0 \end{cases}$$

The Factorial Function

```
def fact(n) do
  if (n > 0) do
    n * fact(n - 1)
  else
    1
  end
end
```

And this is perfectly fine ☺
But we can do better

The Factorial Function

```
def fact(0) do
  1
end
```

The Factorial Function

```
def fact(0) do
  1
end

def fact(n) do
  n * fact(n - 1)
end
```

The Factorial Function

```
def fact(n) do
  n * fact(n - 1)
end

def fact(0) do
  1
end
```

Pattern Matching

```
def process({:ok, result}) do
  result
end

def process({:error, _}) do
  :failure
end

def process(_) do
  :unknown
end
```

Something to Note

```
def fact(0) do
  1
end

def fact(n) do
  n * fact(n - 1)
end
```

Guard Clauses

```
def fact(n) do
  n * fact(n - 1)
end
```

Guard Clauses

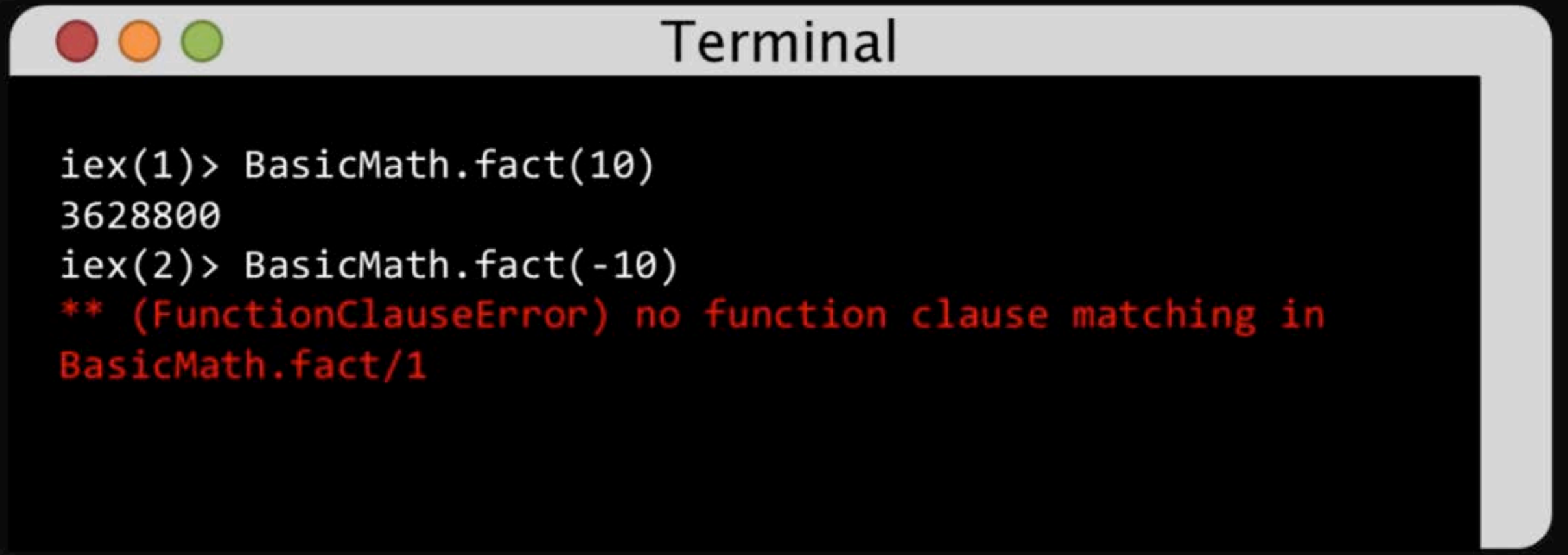
```
def fact(n) when is_integer(n) do
  n * fact(n - 1)
end
```

The Final Factorial Function

```
def fact(0) do
  1
end

def fact(n) when is_integer(n) and n > 0 do
  n * fact(n - 1)
end
```

The Final Factorial Function



Back to the Drawing Board

```
def fact(0) do
  1
end

def fact(n) when is_integer(n) and n > 0 do
  n * fact(n - 1)
end

def fact(_) do
  0
end
```

Finally