09

# Função

### Transcrição

[00:00] Uma função é quando defino um bloco de código que vai ser executado sempre que eu invocar ele. E o método é quando pego essa função e coloco dentro de um objeto, ela precisa de um escopo para poder invocar essa função.

[00:32] Essa definição exatamente pode variar de linguagem para linguagem. Existe um definição teórica que é desnecessária para nós nesse instante. Para a gente, o interessante é saber o que é uma função em Ruby, o que é um método Ruby e o que consigo fazer com eles. Uma função para nós são esses pedaços de código, e não tem prejuízo nenhum em fazer isso. Um método é uma função que existe dentro de um objeto. Podemos chamar se quisermos pensar primeiro o objeto.

[01:11] Estou usando a nomenclatura dessa maneira, apesar de na verdade termos mais ainda do que isso acontecendo. Vamos pegar um exemplo dos nossos testes.rb. Vou definir uma função de bem-vindo. Dentro, vou ter um puts, bem-vindo, e o nome. Se eu parar para pensar que em Ruby tudo é um objeto, então a função será que também não é um objeto?

[01:45] Se a função pudesse ser tratada como um valor, eu poderia dizer que a função é igual a bem-vindo, e depois eu falaria que minha função invoca ela passando meu nome. Mas o que acontece quando rodo esse código?

[02:16] Quando defino com um def uma função, um método, na verdade eu só vou poder invocar ele. Não consigo referenciar só com o nome solto, só bem-vindo. Não posso tratar um método que foi definido dessa maneira como um valor, que posso depois aplicar. Em Ruby não rola. Tem linguagens que permitem, não tem problema. Um exemplo é o Java. Você pode tratar uma função como valor. Mas nosso curso não é sobre Java.

[03:50] No Ruby eu vou fazer o seguinte. Vou falar que o bem-vindo é uma função. A sintaxe muda. Essa função é um objeto em memória. Se ela é um objeto, quando vou invocar, chamo o método call. O método call invoca esse código que está aqui dentro.

[04:56] Estou criando aqui um objeto do tipo proc. Esse objeto tem o método call. Posso referenciar como se fosse uma variável, atribuir para um, para outro, como você quiser. E tem até um atalho com Ctrl para invocar. É feio, mas funciona.

[05:31] Tem situações em que podemos querer tratar uma função como variável, e aí tratamos dessa maneira. No nosso código até agora não precisamos disso, principalmente porque não trabalhamos orientados como uma linguagem funcional. Trabalhos de maneira interativa e usando características de orientação de objeto. Interativo quando a gente ordena, inteira. Mandamos ele fazer coisas. Não estamos preocupados com isso, não é o foco desse curso.

[06:16] Tudo é um objeto que estamos vendo. Todas as funções que estamos criando são na verdade métodos de objetos. Até mesmo esse é o método de uma proc que podemos executar invocando através do call. É tudo um método que estamos invocando, só que por vício de linguagem é natural chamar de função, de objeto de função. Não se preocupe. Ninguém aqui está pegando fogo por isso. Apesar de ser um método de um objeto específico, quando por vício de linguagem dizemos para invocar, sabemos que estamos falando daquele método.

[07:22] Se eu quiser referenciar uma única função e saber que um objeto só tem uma função para invocar, e mesmo assim não é só isso, podemos usar o proc através dessa ou outras estruturas.