

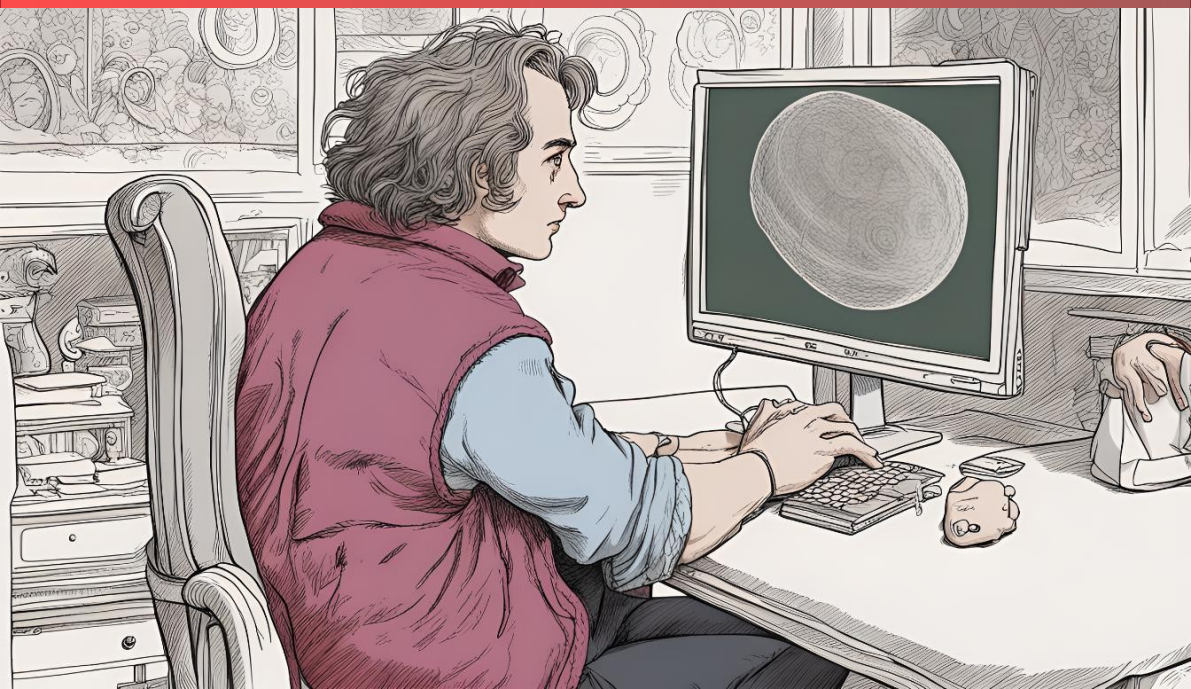
Rubyista das Galáxias



O Guia do Rubyista das Galáxias

Uma Viagem Cósmica na Programação

Dayan Freitas



O Guia do Rubyista das Galáxias

Bem-vindo ao "Guia do Rubyista das Galáxias: Uma Viagem Cósmica na Programação"!

Um ebook criado com inteligência artificial revisado e mantido por mim.

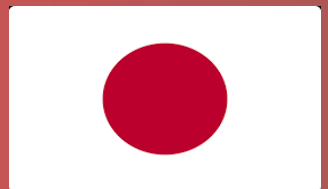
01

Introdução

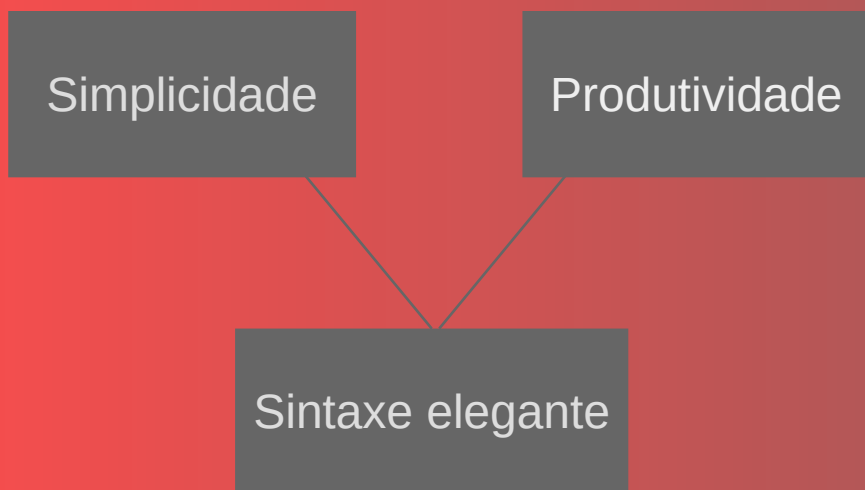
Introdução



Ruby é uma linguagem de programação dinâmica e interpretada, criada em meados da década de 1990 por Yukihiro "Matz" Matsumoto, no Japão.



Introdução



Destacando-se pela sua simplicidade e produtividade, Ruby é conhecida por sua sintaxe elegante e fácil de ler, que permite aos desenvolvedores escreverem menos código sem sacrificar a funcionalidade

Inspirada em linguagens como Perl, Smalltalk, Eiffel e Ada, Ruby foca na felicidade do programador, promovendo princípios de design como a simplicidade, consistência e o uso intuitivo de bibliotecas.

02

Configuração do Ambiente

Configuração do Ambiente

Antes de começarmos nossa jornada pelo vasto universo do *Ruby*, é essencial preparar nosso ambiente de desenvolvimento. Nesta seção, vamos aprender como instalar o *Ruby* em seu computador e configurar um ambiente adequado para explorar todas as maravilhas que esta linguagem tem a oferecer.

Configuração do Ambiente

Para começar a programar em Ruby, é necessário instalar o interpretador Ruby em seu sistema.

Windows: Utilize o instalador RubyInstaller.



Configuração do Ambiente

macOS: Ruby já vem pré-instalado, mas recomenda-se utilizar um gerenciador de versões como `rvm` ou `rbenv`.

Linux: Use o gerenciador de pacotes de sua distribuição (ex.: `sudo apt-get install ruby`).

Windows: Use o `RubyInstaller`.

macOS: Ruby já vem pré-instalado, mas recomenda-se utilizar um gerenciador de

pacotes de sua distribuição (ex.: `sudo apt-get install ruby`).



03

Sintaxe Básica

Sintaxe Básica

Este módulo cobre a sintaxe fundamental de Ruby, incluindo variáveis, tipos de dados, operadores e como escrever comentários. Esses conceitos são a base para entender códigos mais complexos

Sintaxe Básica

Hello World



```
puts 'Hello, world!'
```

Um simples programa que exibe "Hello, world!" no console, ilustrando a simplicidade da sintaxe de Ruby.

Sintaxe Básica

Comentários



```
# Isto é um comentário
```

04

Estruturas de Controle

Estruturas de Controle


Vamos explorar as estruturas de controle de fluxo, como:

Condicionais- If/else/case;
Loops - while/for;

Essas estruturas são essenciais para tomar decisões e repetir ações em seu código.

Estruturas de Controle

If/Else




```
x = 10
if x > 5
  puts 'x é maior que 5'
else
  puts 'x é menor ou igual a 5'
end
```

Este exemplo demonstra uma simples estrutura condicional que verifica se um número é maior que 5.

Estruturas de Controle

Case




```
idade = 18

case idade
when 0..2
  puts "Bebê"
when 3..12
  puts "Criança"
when 13..19
  puts "Adolescente"
else
  puts "Adulto"
end
```

Estruturas de Controle

Loops

```
  
# While  
i = 0  
while i < 5 do  
  puts "i: #{i}"  
  i += 1  
end  
  
# For  
for i in 0..5  
  puts "i: #{i}"  
end
```

Métodos e Funções

Aprenderemos como definir e utilizar métodos em Ruby. Métodos ajudam a organizar o código em blocos reutilizáveis e claros, facilitando a manutenção e a leitura.

Métodos e Funções

Definindo Métodos



```
def soma(a, b)
  a + b
end

resultado = soma(2, 3)
puts resultado # Saída: 5
```

Um método simples que soma dois números e retorna o resultado.

Métodos e Funções

Métodos com Parâmetros Opcionais



```
def saudacao(nome = 'Mundo')  
  "Olá, #{nome}!"  
end  
  
puts saudacao # Saída: Olá, Mundo!  
puts saudacao('João') # Saída: Olá, João!
```

06

Estruturas de Dados

Estruturas de Dados

Este módulo cobre as estruturas de dados fundamentais em Ruby, como arrays e hashes. Estruturas de dados são usadas para armazenar e manipular coleções de informações de forma eficiente.

Estruturas de Dados

Arrays



```
numeros = [1, 2, 3, 4, 5]
numeros.each do |numero|
  puts numero
end
```

Um exemplo de iteração sobre um array, imprimindo cada elemento.

Estruturas de Dados

Hashes



```
peessoa = { nome: 'João', idade: 30 }  
puts peessoa[:nome] # Saída: João
```

07

Programação Orientada a Objetos

Programação Orientada a Objetos

Ruby é uma linguagem totalmente orientada a objetos. Vamos aprender sobre classes, objetos, métodos e atributos, e como usar esses conceitos para modelar problemas do mundo real.

Programação Orientada a Objetos

Classes e Objetos

```
class Pessoa
  attr_accessor :nome, :idade

  def initialize(nome, idade)
    @nome = nome
    @idade = idade
  end

  def saudacao
    "Olá, meu nome é #{@nome} e tenho #{@idade} anos."
  end
end

pessoa = Pessoa.new('Ana', 25)
puts pessoa.saudacao
```

Definindo uma classe Pessoa com métodos e atributos, e criando uma instância dessa classe.

08

Manipulação de Arquivos

Manipulação de Arquivos

Aprenderemos a ler e escrever arquivos em Ruby, uma habilidade importante para tarefas como processamento de dados e armazenamento de informações.

Manipulação de Arquivos

Lendo Arquivos

```
File.open('arquivo.txt', 'r') do |arquivo|  
  while linha = arquivo.gets  
    puts linha  
  end  
end
```

Escrevendo em Arquivos

```
File.open('arquivo.txt', 'w') do |arquivo|  
  arquivo.puts 'Escrevendo no arquivo.'  
end
```

09

Desenvolvimento Web com Ruby on Rails

Ruby on Rails

Este módulo introduz Ruby on Rails, um framework poderoso para desenvolvimento de aplicações web. Veremos como criar um projeto Rails básico e a estrutura de diretórios de um projeto Rails.

Ruby on Rails

Comandos para criar e iniciar um novo projeto Rails, preparando o ambiente para desenvolvimento web.



```
rails new meu_projeto  
cd meu_projeto  
rails server
```

10

Desafios

Desafios

1. Escreva um programa que imprime "Hello, world!" no console.

Dica: Use o método puts.

2. Crie um programa que pede ao usuário dois números e imprime a soma, subtração, multiplicação e divisão desses números.

Dica: Use gets.to_i para capturar a entrada do usuário e convertê-la para um número inteiro.

3. Escreva um programa que solicita um número ao usuário e informa se o número é par ou ímpar.

Dica: Use o operador % para verificar a paridade.

4. Crie um programa que faz uma contagem regressiva de 10 até 1 e depois imprime "Feliz Ano Novo!".

Dica: Use um loop while ou downto.

5. Escreva um programa que converte uma temperatura de Celsius para Fahrenheit.

Fórmula: $\text{Fahrenheit} = (\text{Celsius} * 9/5) + 32$

Rubyista das Galáxias

O Guia do Rubyista das Galáxias

Uma Viagem Cósmica na Programação

Dayan Freitas