

DOJOMARINGÁ

Ruby: Documentação e variáveis

DOCUMENTAÇÃO

Ferramenta: Ruby Information

“

Grande parte delas são elaboradas com **RDoc**, onde são apenas comentários que podem ser extraídos para HTML ou para o formato **ri**.

DOCUMENTAÇÃO

Ferramenta: Ruby Information

- ▶ Classes
- ▶ Constantes
- ▶ Módulos
- ▶ Métodos

DOCUMENTAÇÃO

Ferramenta: Ruby Information

► Classes, constantes e módulos

```
$ ri Array #=> Nome da classe
```

```
$ ri ENV #=> Nome da constante
```

```
$ ri Enumerable #=> Nome do módulo
```

DOCUMENTAÇÃO

Ferramenta: Ruby Information

- Classes, constantes e módulos



```
ruby
= Array < Object

-----
= Includes:
Enumerable (from ruby core)
(from ruby core)
-----
Arrays are ordered, integer-indexed collections of any object. Array indexing
starts at 0, as in C or Java. A negative index is assumed to be relative to
the end of the array---that is, an index of -1 indicates the last element of
the array, -2 is the next to last element in the array, and so on.
-----
= Class methods:

[], new, try_convert

= Instance methods:

&, *, +, -, <<, <=>, ==, [], []=, abbrev, assoc, at, clear, collect,
collect!, combination, compact, compact!, concat, count, cycle, dclone,
delete, delete_at, delete_if, drop, drop_while, each, each_index, empty?,
eql?, fetch, fill, find_index, first, flatten, flatten!, frozen?, hash,
include?, index, initialize_copy, insert, inspect, join, keep_if, last,
length, map, map!, pack, permutation, pop, pretty_print, pretty_print_cycle,
:|
```

DOCUMENTAÇÃO

Ferramenta: Ruby Information

► Métodos

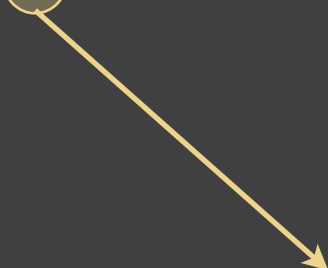
\$ ri Hash#each #=> Nome da classe e nome do método

DOCUMENTAÇÃO

Ferramenta: Ruby Information

► Métodos

\$ ri Hash#each #=> Nome da classe e nome do método



Símbolo para
especificar um
método da classe

DOCUMENTAÇÃO

Ferramenta: Ruby Information

► Métodos



```
ruby
= Hash#each

(from ruby core)

-----
hsh.each      { |key, value| block } -> hsh
hsh.each_pair { |key, value| block } -> hsh
hsh.each      -> an_enumerator
hsh.each_pair -> an_enumerator
-----

Calls block once for each key in hsh, passing the key-value
pair as parameters.

If no block is given, an enumerator is returned instead.

h = { "a" => 100, "b" => 200 }
h.each { |key, value| puts "#{key} is #{value}" }

produces:

a is 100
b is 200
:|
```


DOCUMENTAÇÃO

Ferramenta: Ruby Information

EXEMPLOS

Rápidos exemplos usando o Ruby Information para acessar classes, métodos, constantes e módulos diversos da linguagem.

VARIÁVEIS

Definições: Entendendo seus valores

“

*Existem maneiras para definir variáveis no Ruby. Em geral, são definidas com nomes iniciados com uma letra minúscula e underscore para alguma composição (**snake case**).*

VARIÁVEIS

Definições: Entendendo seus valores

- ▶ Variáveis locais
- ▶ Variáveis globais
- ▶ Variáveis de instância
- ▶ Variáveis de classe

VARIÁVEIS

Definições: Entendendo seus valores

► Variáveis locais

Em escopo local, elas não possuem atribuição do valor **nil** antes da inicialização, ao contrário das variáveis globais e as variáveis de instância.

```
$ irb
```

```
irb(main):001:0> nome = "John Doe"
```

```
=> "John Doe"
```

```
irb(main):002:0> defined? nome
```

```
=> "local-variable"
```

Verificação de escopo:

- Proc
- Loop
- Método
- Classe
- Módulo
- Aplicação

VARIÁVEIS

Definições: Entendendo seus valores

► Variáveis locais

Proc objects são blocos de código vinculados a variáveis locais. Uma *proc*, ou *procedure*, tem seu próprio escopo e pode ser chamado várias vezes em contextos diferentes.

CURIOSIDADE

VARIÁVEIS

Definições: Entendendo seus valores

► Variáveis globais

O conceito de variável global é, basicamente, uma referência acessível em qualquer parte da aplicação. Existe uma convenção de nomes onde diz que estas variáveis devem ser iniciadas com **\$** (cifrão).

```
$ irb
```

```
irb(main):001:0> $versao = "10.5"
```

```
=> "10.5"
```

```
irb(main):002:0> defined? $versao
```

```
=> "global-variable"
```

VARIÁVEIS

Definições: Entendendo seus valores

► Variáveis de instância

Estas variáveis implementam os atributos de uma classe e, coletivamente, representam o estado de um objeto. São definidas de forma independente dos outros objetos da mesma classe.

```
class Animal
  attr_accessor :nome, :tipo

  def initialize(nome, tipo)
    self.nome, self.tipo = nome, tipo
  end
end
```

```
class Passaro < Animal
  def initialize(nome, tipo)
    super
  end
end
```

```
$ irb

irb(main):001:0> p = Passaro.new("Test", "Test")
=> #<Passaro:0x001234567 @nome="Test", @tipo="Test">
irb(main):002:0> p.instance_variables
=> [:@nome, @tipo]
```

VARIÁVEIS

Definições: Entendendo seus valores

► Variáveis de classe

Estas variáveis são responsáveis por guardar informações de um classe. São compartilhadas entre todos os objetos da classe em questão. Variáveis de classe devem ser iniciadas com @@ (arroba arroba).

```
class Animal
  @@caracteristica = "voa"

  def self.caracteristica
    @@caracteristica
  end
end

class Pato < Animal
  @@caracteristica = "anda"
end
```

```
$ irb

irb(main):001:0> Animal.caracteristica
=> "anda"
irb(main):002:0> Pato.caracteristica
=> "anda"
irb(main):003:0> Animal.class_variables
=> [:@@caracteristica]
```


VARIÁVEIS

Definições: Entendendo seus valores

EXEMPLOS

Rápidos exemplos de definição de variáveis em seus mais diferentes escopos na linguagem.

PROBLEM?

Perguntas?



OBRIGADO!

@rogeriozambon