String

© 2022 By Sun* - Talent Incubator Vietnam Unit - All rights reserved

Outline

1. Introduction
2. String interpolation
3. Ruby accessing string elements
4. Ruby concatenating strings
5. Ruby comparing strings
6. String methods

© 2022 By Sun* - Talent Incubator Vietnam Unit - All rights reserved

1. Introduction

- String are one of the most important data types in computer languages.
- A string is a sequence of characters.
- String objects may be created using **String.new**. When a string appears literally in source code, it is known as a *string literal*.
- In Ruby, string literals are enclosed by single or double quotes.

© 2022 By Sun* - Talent Incubator Vietnam Unit - All rights reserved

1. Introduction

```
p "the quick brown fox jumps over the lazy dog"
p 'the quick brown fox jumps over the lazy dog'
p 'the quick brown fox jumps over the lazy dog'.class
```

```
#Result
"the quick brown fox jumps over the lazy dog"
"the quick brown fox jumps over the lazy dog"
String
```

© 2022 By Sun* - Talent Incubator Vietnam Unit - All rights reserved

2. String interpolation

```
p "the quick brown " + "fox" + "jumps over the lazy " + "dog"

# string interpolation
puts "Enter name of an animal"
animal = gets.chomp
puts "Enter a noun"
noun = gets.chomp
p "the quick brown #{animal} jumps over the lazy #{noun}"
# try again with single quote
p 'the quick brown #{animal} jumps over the lazy #{noun}'

# Other example
p "the quick brown #{2 + 2}"
```

© 2022 By Sun* - Talent Incubator Vietnam Unit - All rights reserved

3. Ruby accessing string elements

- It is possible to access string elements in Ruby.
- For this we use the square brackets []. Inside the brackets, we can put strings, indexes, or ranges..

```
2.7.1 :001 > msg = "Ruby language"
 => "Ruby language"
2.7.1 :002 > puts msg["Ruby"]
 => Ruby
2.7.1 :003 > puts msg[0]
 => "R"
2.7.1 :004 > puts msg[0..3]
 => "Ruby"
2.7.1 :005 > puts msg[-1]
 => "e"
2.7.1 :006 > puts msg[5] = "L"
 => "L"
```

© 2022 By Sun* - Talent Incubator Vietnam Unit - All rights reserved

4. Ruby concatenating strings

- Concatenating strings is creating one string from multiple strings.

```
2.7.1 :001 > "Ruby" + " programming" + " language"
 => "Ruby programming language"

2.7.1 :002 > "Ruby" " programming" " language"
 => "Ruby programming language"

2.7.1 :003 > "Ruby" << " programming" << " language"
 => "Ruby programming language"

2.7.1 :004 > "Ruby".concat(" programming").concat(" language")
 => "Ruby programming language"
```

© 2022 By Sun* - Talent Incubator Vietnam Unit - All rights reserved

5. Ruby comparing strings

- We can compare two strings with a == operator or with a eql? method. They return true if the strings are equal and false if not

```
2.7.1 :001 > "12" == "12"
 => true

2.7.1 :002 > "aa" == "ab"
 => false

2.7.1 :003 > "Jane".eql? "Jan"
 => false

2.7.1 :004 > "Jane".eql? "Jane"
 => true
```

© 2022 By Sun* - Talent Incubator Vietnam Unit - All rights reserved

6. String methods

@anonymous/BigheartedCostlyWhitepelican
No description

Log in

saved

```
1 String.public_instance_methods.sort
2
```

input clear

```
ruby 2.3.1p112 (2016-04-26 revision 54768) [x86_64-linux]
>
=> [:!, :!=, :!~, :%, :*, :+, :+@, :-@, :<, :<<, :<=, :<=>, :==, :===, :=~,
:>, :>=, :[], :[]=, :__id__, :__send__, :ascii_only?, :b, :between?, :bytes,
:bytesize, :byteslice, :capitalize, :capitalize!, :casecmp, :center, :chars,
:chomp, :chomp!, :chop, :chop!, :chr, :class, :clear, :clone, :codepoints,
:concat, :count, :crypt, :define_singleton_method, :delete, :delete!,
:display, :downcase, :downcase!, :dump, :dup, :each_byte, :each_char,
:each_codepoint, :each_line, :empty?, :encode, :encode!, :encoding,
:end_with?, :enum_for, :eql?, :equal?, :extend, :force_encoding, :freeze,
:frozen?, :getbyte, :gsub, :gsub!, :hash, :hex, :include?, :index, :insert,
:inspect, :instance_eval, :instance_exec, :instance_of?,
:instance_variable_defined?, :instance_variable_get, :instance_variable_set,
:instance_variables, :intern, :is_a?, :itself, :kind_of?, :length, :lines,
:ljust, :lstrip, :lstrip!, :match, :method, :methods, :next, :next!, :nil?,
:object_id, :oct, :ord, :partition, :prepend, :private_methods,
:protected_methods, :public_method, :public_methods, :public_send,
:remove_instance_variable, :replace, :respond_to?, :reverse, :reverse!,
:rindex, :rjust, :rpartition, :rstrip, :rstrip!, :scan, :scrub, :scrub!,
:send, :setbyte, :shellescape, :shellsplit, :singleton_class,
:singleton_method, :singleton_methods, :size, :slice, :slice!, :split,
:squeeze, :squeeze!, :start_with?, :strip, :strip!, :sub, :sub!, :succ,
:succ!, :sum, :swapcase, :swapcase!, :taint, :tainted?, :tap, :to_c,
:to_enum, :to_f, :to_i, :to_json, :to_json_raw, :to_json_raw_object, :to_r,
:to_s, :to_str, :to_sym, :tr, :tr!, :tr_s, :tr_s!, :trust,
:unicode_normalize, :unicode_normalize!, :unicode_normalized?, :unpack,
:untaint, :untrust, :untrusted?, :upcase, :upcase!, :upto, :valid_encoding?]
>
```

© 2022 By Sun* - Talent Incubator Vietnam Unit - All rights reserved

6. String methods

https://ruby-doc.org/core-3.0.1/String.html

Home Classes Methods

In Files

- complex.c
- pack.c
- rational.c
- string.c
- transcode.c

Parent

Object

Methods

::new
::try_convert
#%
#*
#+
#+@
#-@
#<<
#<=>
#==

String

A `String` object holds and manipulates an arbitrary sequence of bytes, typically representing characters. String objects may be created using `String::new` or as literals.

Because of aliasing issues, users of strings should be aware of the methods that modify the contents of a `String` object. Typically, methods with names ending in "!" modify their receiver, while those without a "!" return a new `String`. However, there are exceptions, such as `String#[]=`.

Public Class Methods

new(str="") → new_str
new(str="", encoding: enc) → new_str
new(str="", capacity: size) → new_str

Returns a new string object containing a copy of *str*.

The optional *enc* argument specifies the encoding of the new string. If not specified, the encoding of *str* (or ASCII-8BIT, if *str* is not specified) is used.

© 2022 By Sun* - Talent Incubator Vietnam Unit - All rights reserved

6. String methods

```
p "###Upcase"
p "The quick brown fox jumps over the lazy dog".upcase
p "###Downcase"
p "The quick brown fox jumps over the lazy dog".downcase
p "###Swapcase"
p "The quick brown fox jumps over the lazy dog".swapcase
p "###Reverse"
p "The quick brown fox jumps over the lazy dog".reverse
p "###Reverse-Upcase"
p "The quick brown fox jumps over the lazy dog".reverse.upcase
```

```
#Result
"###Upcase"
"THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG"
"###Downcase"
"the quick brown fox jumps over the lazy dog"
"###Swapcase"
"tHE QUICK BROWN FOX JUMPS OVER THE LAZY DOG"
"###Reverse"
"god yzal eht revo spmuj xof nworb kciuq ehT"
"###Reverse-Upcase"
"GOD YZAL EHT REVO SPMUJ XOF NWORB KCIUQ EHT"
```

© 2022 By Sun* - Talent Incubator Vietnam Unit - All rights reserved

6. String methods: gsub

- **gsub** method: **gsub(**pattern, replacement**) or gsub(**pattern**){**|match| block**}**: return a copy of string with the all occurrence of pattern replaced by the second argument.

```
2.7.1 :032 > s = "the quick brown fox jumps over the lazy dog"
 => "the quick brown fox jumps over the lazy dog"
2.7.1 :033 > s.gsub(/[aeiou]/, '*')
 => "th* q**ck br*wn f*x j*mps *v*r th* l*zy d*g"
2.7.1 :034 > s.gsub('e', '*')
 => "th* quick brown fox jumps ov*r th* lazy dog"
2.7.1 :035 > s.gsub('e') {|c| c.ord.to_s}
 => "th101 quick brown fox jumps ov101r th101 lazy dog"
2.7.1 :038 > s.gsub(/[eo]/, 'e' => 8, 'o' => 9)
 => "th8 quick br9wn f9x jumps 9vr th8 lazy d9g"
```

© 2022 By Sun* - Talent Incubator Vietnam Unit - All rights reserved

6. String methods: gsub!

- **gsub!** method: return string if a substitution was performed or *nil* if no.

```
2.7.1 :047 >   s = "the quick brown fox jumps over the lazy dog"
 => "the quick brown fox jumps over the lazy dog"
2.7.1 :048 > s.gsub!(/[aeiou]/, '*')
 => "th* q**ck br*wn f*x j*mps *v*r th* l*zy d*g"
2.7.1 :049 > s.gsub!('e', '*')
 => nil
2.7.1 :050 > s
 => "th* q**ck br*wn f*x j*mps *v*r th* l*zy d*g"
```

© 2022 By Sun* - Talent Incubator Vietnam Unit - All rights reserved

6. String methods: split

- **split** method**:** divides *str* into substrings based on a delimiter, returning an array of these substrings.

```
2.7.1 :091 >   s = "the quick brown fox jumps over the lazy dog"
 => "the quick brown fox jumps over the lazy dog"
2.7.1 :092 > s.split
 => ["the", "quick", "brown", "fox", "jumps", "over", "the", "lazy", "dog"]
2.7.1 :093 > s.split(' ')
 => ["the", "quick", "brown", "fox", "jumps", "over", "the", "lazy", "dog"]
2.7.1 :094 > s.split(/ /)
 => ["the", "quick", "brown", "fox", "jumps", "over", "the", "lazy", "dog"]
2.7.1 :095 > s1 = " the   quick brown fox jumps over the lazy dog "
 => " the   quick brown fox jumps over the lazy dog "
2.7.1 :096 >       s1.split(/ /)
 => ["", "the", "", "", "quick", "brown", "fox", "jumps", "over", "the", "lazy", "dog"]
```

© 2022 By Sun* - Talent Incubator Vietnam Unit - All rights reserved

6. String methods: split

```
2.7.1 :097 > s1.split(' ', 1)
 => [" the   quick brown fox jumps over the lazy dog "]
2.7.1 :098 > s1.split(' ', 4)
 => ["the", "quick", "brown", "fox jumps over the lazy dog "]
2.7.1 :099 > s1.split(' ', 5)
 => ["the", "quick", "brown", "fox", "jumps over the lazy dog "]
2.7.1 :100 > s1.split(' ', -5)
 => ["the", "quick", "brown", "fox", "jumps", "over", "the", "lazy", "dog", ""]
2.7.1 :101 > s1.split(' ', -1)
 => ["the", "quick", "brown", "fox", "jumps", "over", "the", "lazy", "dog", ""]
2.7.1 :102 > "".split
 => []
2.7.1 :103 > "".split(',', 3)
 => []
```

© 2022 By Sun* - Talent Incubator Vietnam Unit - All rights reserved

6. String methods: strip

- **strip** method: returns a copy of str with leading and trailing whitespace removed.

```
2.7.1 :127 > s = "\t  \r the quick brown fox jumps over the lazy dog         "
=> "\t  \r the quick brown fox jumps over the lazy dog         "
2.7.1 :128 > s.strip
 => "the quick brown fox jumps over the lazy dog"
2.7.1 :129 > s1 = "\t  \r the   quick brown fox jumps over the lazy dog      \n       "
 => "\t  \r the       quick brown fox jumps over the lazy dog         "
2.7.1 :130 > s1.strip
 => "the       quick brown fox jumps over the lazy dog"
```

© 2022 By Sun* - Talent Incubator Vietnam Unit - All rights reserved

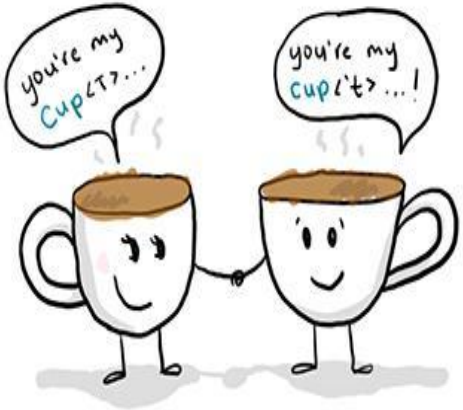
References

- http://zetcode.com/lang/rubytutorial/strings/
- https://ruby-doc.org/core-3.1.0/String.html
- https://github.com/awesome-academy/RubyExample_TFW

© 2022 By Sun* - Talent Incubator Vietnam Unit - All rights reserved

Question & Answer?

© 2022 By Sun* - Talent Incubator Vietnam Unit - All rights reserved



© 2022 By Sun* - Talent Incubator Vietnam Unit - All rights reserved