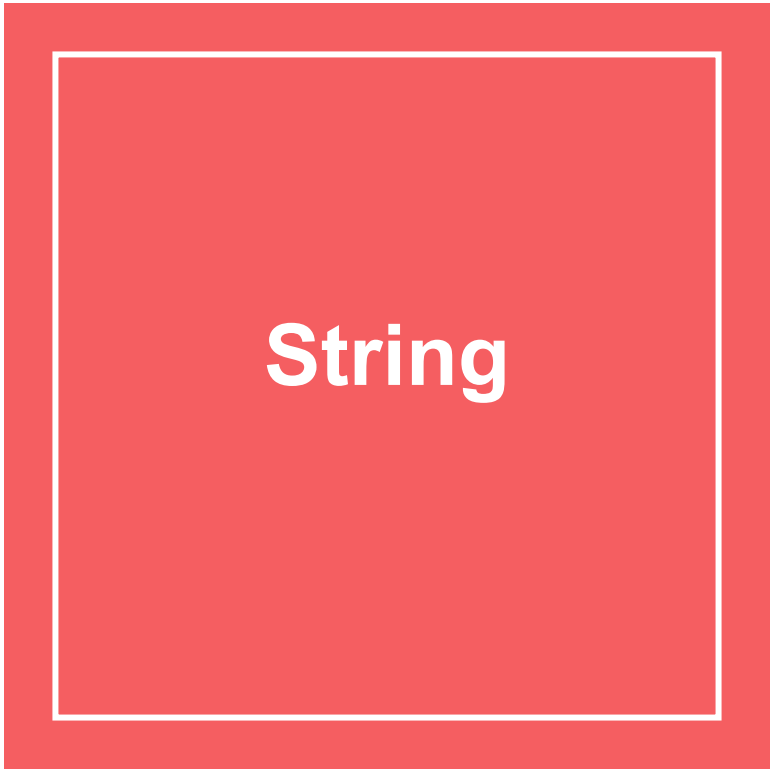


Outline

1. Introduction
2. String interpolation
3. String manipulation
4. String substitution
5. Split and Strip methods

1. Introduction

- String are one of the most important data types in computer languages.
- A string is a sequence of Unicode characters. It is a data type that stores a sequence of data values in which elements usually stand for characters according to a character encoding.
- String objects may be created using **String::new** or as *literals.* 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.

1. Introduction

```
2.4.0 :001 > the quick brown fox jumps over the lazy dog
NameError: undefined local variable or method `dog' for main:Object

2.4.0 :001 > "the quick brown fox jumps over the lazy dog"
"the quick brown fox jumps over the lazy dog"

2.4.0 :002 > 'the quick brown fox jumps over the lazy dog'
"the quick brown fox jumps over the lazy dog"

2.4.0 :003 > 'the quick brown fox jumps over the lazy dog'.class
String < Object
```

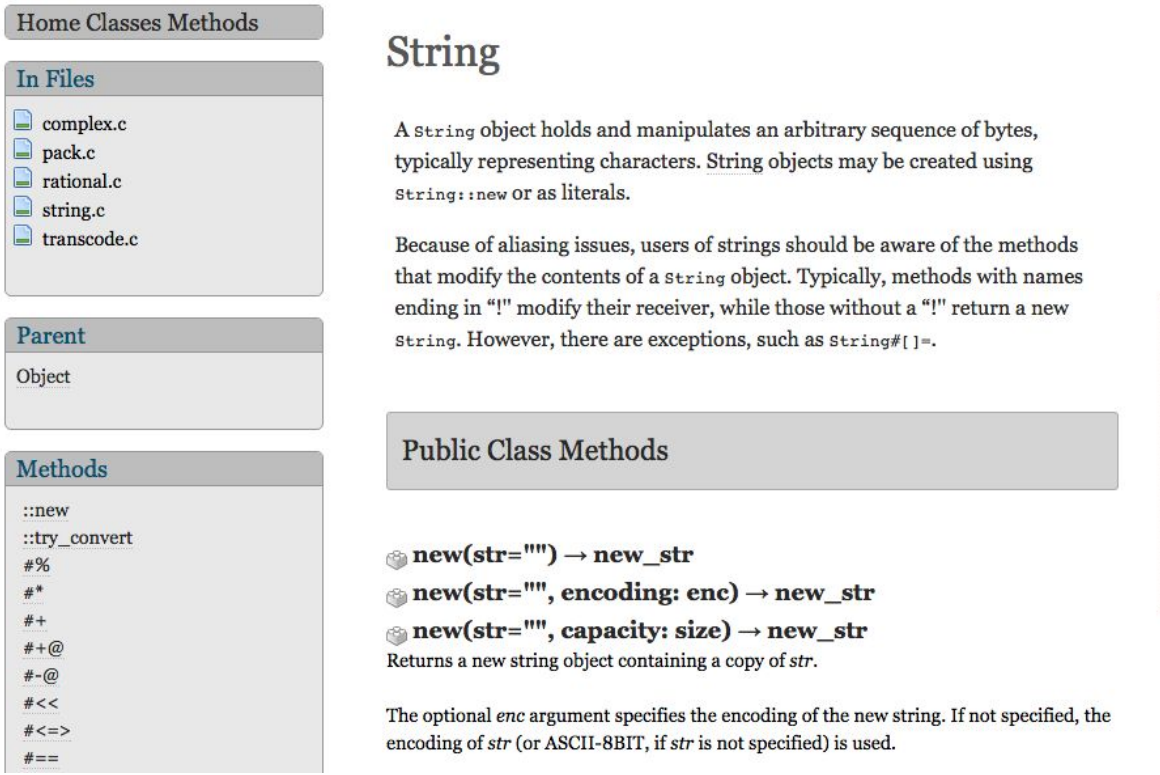
2. String interpolation

```
# concatenating strings with +
2.4.0 :004 > "the quick brown " + "fox" + "jumps over the lazy " + "dog"
"the quick brown foxjumps over the lazy dog"

# string interpolation
puts "Enter name 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}"
```

3. String manipulation

```
2.4.0 :030 > "The quick brown fox jumps over the lazy dog".upcase
"THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG"

2.4.0 :031 > "The quick brown fox jumps over the lazy dog".downcase
"the quick brown fox jumps over the lazy dog"

2.4.0 :032 > "The quick brown fox jumps over the lazy dog".swapcase
"tHE QUICK BROWN FOX JUMPS OVER THE LAZY DOG"

2.4.0 :033 > "The quick brown fox jumps over the lazy dog".reverse
"god yzal eht revo spmuj xof nworb kciuq ehT"

2.4.0 :034 > "The quick brown fox jumps over the lazy dog".reverse.upcase
"GOD YZAL EHT REVO SPMUJ XOF NWORB KCIUQ EHT"
```

3. String manipulation



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.

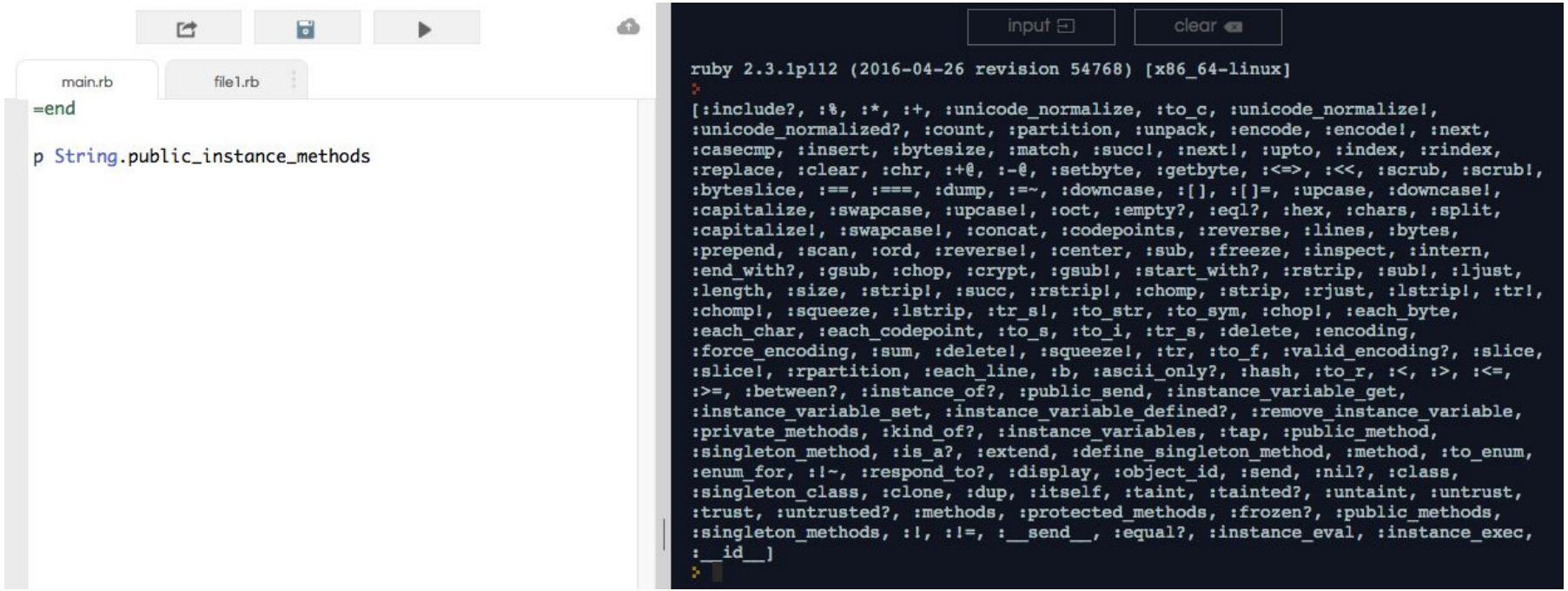
Because of aliasing issues, users of strings should be aware of the methods that modify the contents of a `String` object. Typically, methods with names ending in "!" modify their receiver, while those without a "!" return a new `String`. However, there are exceptions, such as `String#[]=`.

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.

3. String manipulation



4. String substitution

- **sub** method

sub(pattern, replacement**) or sub(**pattern**) {**|match| block**}**: return a copy of string with the first occurrence of pattern replaced by the second argument (the pattern is typically a Regexp)

```
2.4.0 :001 > s = "the quick brown fox jumps over the lazy dog"
 => "the quick brown fox jumps over the lazy dog"
2.4.0 :002 > s.sub(/[aeiou]/, '*')
 => "th* quick brown fox jumps over the lazy dog"
2.4.0 :003 > s.sub('e', '*')
 => "th* quick brown fox jumps over the lazy dog"
2.4.0 :004 > s.sub(/./) {|c| c.ord.to_s}
 => "116he quick brown fox jumps over the lazy dog"
```

note**:** if given as a String, any regular expression metacharacters it contains will be interpreted literally, e.g. '\\d' will match a backlash followed by ‘d’, instead of a digit

4. String substitution

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

```
2.4.0 :012 > s.sub!(/[aeiou]/, '*')
 => "th* quick brown fox jumps over the lazy dog"
2.4.0 :013 > s
 => "th* quick brown fox jumps over the lazy dog"
2.4.0 :014 > s.sub!(/./) {|c| c.ord.to_s}
 => "116h* quick brown fox jumps over the lazy dog"
2.4.0 :015 > s
 => "th* q**ck br*wn f*x j*mps *v*r th* l*zy d*g"
2.4.0 :029 > s.sub!(/[aeiou]/, '*')
 => nil
```

4. String substitution

- **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

```
2.4.0 :032 > s = "the quick brown fox jumps over the lazy dog"
 => "the quick brown fox jumps over the lazy dog"
2.4.0 :033 > s.gsub(/[aeiou]/, '*')
 => "th* q**ck br*wn f*x j*mps *v*r th* l*zy d*g"
2.4.0 :034 > s.gsub('e', '*')
 => "th* quick brown fox jumps ov*r th* lazy dog"
2.4.0 :035 > s.gsub('e') {|c| c.ord.to_s}
 => "th101 quick brown fox jumps ov101r th101 lazy dog"
2.4.0 :038 > s.gsub(/[eo]/, 'e' => 8, 'o' => 9)
 => "th8 quick br9wn f9x jumps 9vr th8 lazy d9g"
```

4. String substitution

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

```
2.4.0 :047 >   s = "the quick brown fox jumps over the lazy dog"
 => "the quick brown fox jumps over the lazy dog"
2.4.0 :048 > s.gsub!(/[aeiou]/, '*')
 => "th* q**ck br*wn f*x j*mps *v*r th* l*zy d*g"
2.4.0 :049 > s.gsub!('e', '*')
 => nil
2.4.0 :050 > s
 => "th* q**ck br*wn f*x j*mps *v*r th* l*zy d*g"
```

5. Split and Strip methods

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

```
2.4.0 :091 >   s = "the quick brown fox jumps over the lazy dog"
 => "the quick brown fox jumps over the lazy dog"
2.4.0 :092 > s.split
 => ["the", "quick", "brown", "fox", "jumps", "over", "the", "lazy", "dog"]
2.4.0 :093 > s.split(' ')
 => ["the", "quick", "brown", "fox", "jumps", "over", "the", "lazy", "dog"]
2.4.0 :094 > s.split(/ /)
 => ["the", "quick", "brown", "fox", "jumps", "over", "the", "lazy", "dog"]
2.4.0 :095 > s1 = " the   quick brown fox jumps over the lazy dog "
 => " the   quick brown fox jumps over the lazy dog "
2.4.0 :096 > s1.split(/ /)
 => ["", "the", "", "", "quick", "brown", "fox", "jumps", "over", "the", "lazy", "dog"]
```

5. Split and Strip methods

```
2.4.0 :097 > s1.split(' ', 1)
 => [" the   quick brown fox jumps over the lazy dog "]
2.4.0 :098 > s1.split(' ', 4)
 => ["the", "quick", "brown", "fox jumps over the lazy dog "]
2.4.0 :099 > s1.split(' ', 5)
 => ["the", "quick", "brown", "fox", "jumps over the lazy dog "]
2.4.0 :100 > s1.split(' ', -5)
 => ["the", "quick", "brown", "fox", "jumps", "over", "the", "lazy", "dog", ""]
2.4.0 :101 > s1.split(' ', -1)
 => ["the", "quick", "brown", "fox", "jumps", "over", "the", "lazy", "dog", ""]
2.4.0 :102 > "".split
 => []
2.4.0 :103 > "".split(',', 3)
 => []
```

5. Split and Strip methods

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

```
2.4.0 :127 > s = "\t  \r the quick brown fox jumps over the lazy dog        "
=> "\t  \r the quick brown fox jumps over the lazy dog        "
2.4.0 :128 > s.strip
 => "the quick brown fox jumps over the lazy dog"
2.4.0 :129 > s1 = "\t  \r the       quick brown fox jumps over the lazy dog        "
 => "\t  \r the       quick brown fox jumps over the lazy dog        "
2.4.0 :130 > s1.strip
 => "the       quick brown fox jumps over the lazy dog"
```

References

- http://zetcode.com/lang/rubytutorial/strings/
- http://ruby-doc.org/core-2.4.1/String.html

Thank you!