visuality

MORE Ruby, less Rails

Rediscover beauty of Ruby

Ruby Gems

Package of Ruby code that provides a specific functionality.

Stdlib
Default Gems
Bundled Gems
Native Extensions
Independent Gems

https://stdgems.org/

https://alchemists.io/articles/ruby_default_gems

https://docs.ruby-lang.org/en/3.4/standard_library_md.html

Stdlib

Installed with Ruby

Not packed as a separate .gem files

Can’t be updated independently of Ruby

Math
Object
Socket
MakeMakefile
Gem

Default gems

Installed with Ruby

Can’t be removed

Maintained by Ruby core

benchmark
did_you_mean
irb
singleton
prettyprint

Bundled gems

Installed with Ruby

Can be removed or updated

Maintained outside of Ruby core

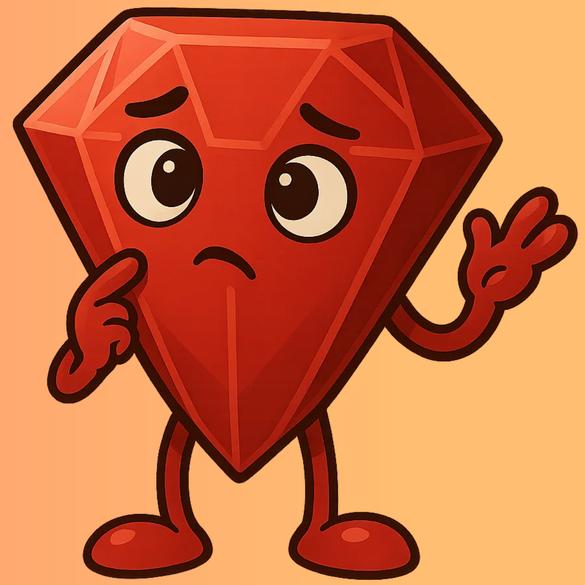
abbrev
csv
minitest
prime
rake

Native extensions

Require compilation on install

Platform dependent

pg
nokogiri
mysql2



Ruby Heredocs

```
str = <<EOF
This is a
multi-line string
EOF

# str
"This is a
multi-line string"
```

```
str = <<~"TXT"
  Baltic
  Ruby
TXT

puts str
# prints:

"Baltic
Ruby"
```

Squiggly heredoc << ~

```
str = <<-EOF
  Baltic
  Ruby
    EOF

puts str
# prints:

"  Baltic
   Ruby"
```

<< -

Ruby Heredocs

Starting tag and closing tag must match

String interpolation turned on by default

<<~ strips indendation

<< - strips indendation of closing tag

https://blog.saeloun.com/2020/04/08/heredoc-in-ruby-and-rails/

Ruby Refinements

1. `Refine` object in a module

```ruby
require 'date'

module Refinements
  refine Date do
    def to_s
      strftime("%A, %B %d, %Y")
    end
  end
end
```

2. Include the module with `using`

```ruby
class ReportGenerator
  using Refinements

  def print_date
    puts Date.new(2025, 06, 12).to_s
  end
end

ReportGenerator.new.print_date
# "Thursday, June 12, 2025"
```

Ruby Refinements 💎

- **You can refine class methods (`refine Date.singleton_class`)**
- **You can't refine constants, nor class variables**
- **Monkey Patching a refinement will void the refinement**

https://alchemists.io/articles/ruby_refinements

https://github.com/bkuhlmann/refinements

Special Variables

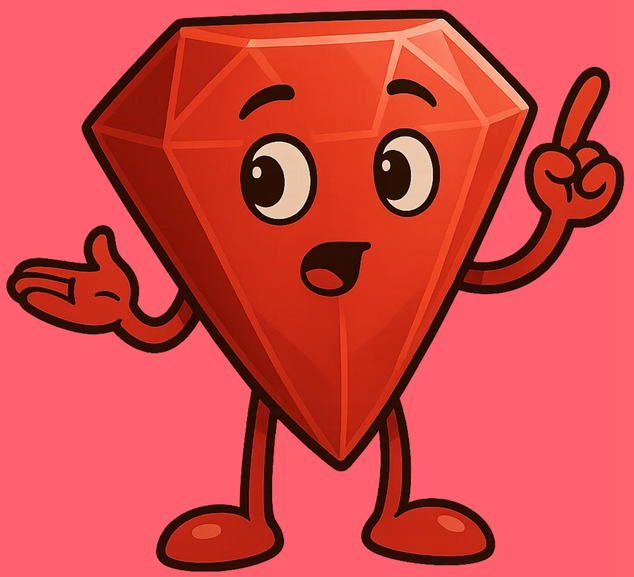
Global variables
automatically
set up by Ruby

```ruby
begin
  raise "error!"
rescue
  puts "Error raised: #{$!}"
end

"Error raised: error!"
# ⇒ nil
```

$= - case-insensitivity flag
$/ - input record separator
$\ - output record separator
$0 - the name of the Ruby script file
$* - the command line arguments
$$ - interpreter's process ID
$? - exit status of last executed child process
$! - latest error message
$@ - location of error
$_ - string last read by `gets`
$. - line number last read by interpreter
$& - string last matched by regexp
$~ - the last regexp match, as an array of subexpressions
$1, $2, ..., $n - the nth subexpression in the last match (same as $~[n])

Ensure

Use `return` if you need `ensure` block to return a value!



```ruby
def always_run
  begin
    "in progress"
    raise "oops"
  ensure
    return "Done"
  end
end

irb(main):133> result = always_run
# (irb):125:in `always_run': oops (RuntimeError)
#       from (irb):130:in `<main>'
#     from <internal:kernel>:187:in `loop'
irb(main):134> result
⇒ "Done"
```

Reserved keywords

Words that are part of the language syntax.
You cannot use them as variable, method or class/module names.

Enumerable#tally

```
["a", "b", "a", "c", "b", "a"].tally
# returns
{"a"⇒3, "b"⇒2, "c"⇒1}

[].tally
# returns
{}
```

Counts elements in a collection

```
{foo: 'a', bar: 'b'}.tally
# returns
{[:foo, "a"]⇒1, [:bar, "b"]⇒1}
```

Can be invoked on Hash

```
h = %w[a c d b c a].tally
# returns
{"a"⇒2, "c"⇒2, "d"⇒1, "b"⇒1}

%w[b a z].tally(h)
# returns
{"a"⇒3, "c"⇒2, "d"⇒1, "b"⇒2, "z"⇒1}
```

Adds counts to given Hash

```ruby
[1, 2, 3].all? { |n| n > 0 }
# returns
true

[].all? { |n| n > 0 }
# returns
true
```

Enumerable#all?

```ruby
[1, 2, 3].none? { |n| n > 5 }
# returns
true

[].none? { |n| n > 0 }
# returns
true
```

Enumerable#none?

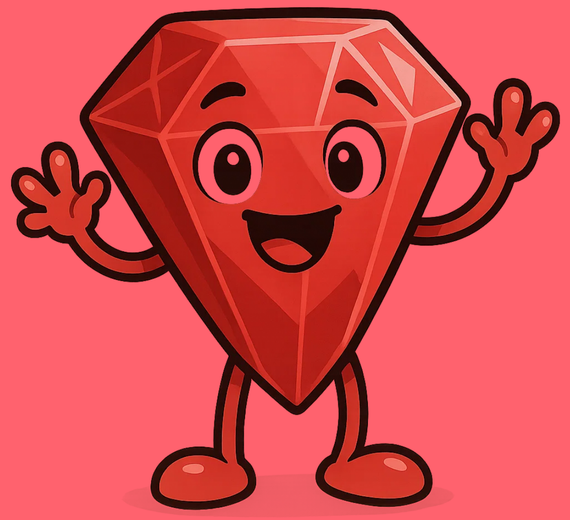
```ruby
[1, 2, 3].any? { |n| n > 2 }
# returns
true

[].any? { |n| n > 0 }
# returns
false
```

Enumerable#any?

Enumerable#each_slice

```
(1..6).each_slice(2) { |slice| p slice }
[1, 2]
[3, 4]
[5, 6]
```

```
(1..7).each_slice(2) { |slice| p slice }
[1, 2]
[3, 4]
[5, 6]
[7]
```



Be aware of collections which can’t be split evenly!

```ruby
words = %w[BEGIN foo bar BEGIN baz]
words.slice_before("BEGIN").to_a
# returns
[["BEGIN", "foo", "bar"], ["BEGIN", "baz"]]
```

Enumerable#slice_before

Enumerable#slice_after

```ruby
words = %w[foo bar END baz qux END]
words.slice_after("END").to_a
# returns
[["foo", "bar", "END"], ["baz", "qux", "END"]]
```

```ruby
[1, 2, 4, 5, 10].slice_when { |a, b| b - a > 1 }.to_a
# returns
[[1, 2], [4, 5], [10]]
```

Enumerable#slice_when

Enumerable#zip

```ruby
[1, 2, 3].zip(["a", "b", "c"])
# returns
[[1, "a"], [2, "b"], [3, "c"]]
```

```ruby
[1, 2, 3].zip(["a", "b"])
# returns
[[1, "a"], [2, "b"], [3, nil]]
```

```ruby
[1, 2].zip(["a", "b", "c"])
# returns
[[1, "a"], [2, "b"]]
```

Ruby Association

Ruby Certification Exam

- Silver and Gold level
- 50 multiple-choice questions
- 90 minutes to complete
- Onsite in Prometric examination center

https://github.com/ruby-association/prep-test/blob/master/silver.md

https://www.visuality.pl/posts/how-to-become-a-ruby-certified-programmer

https://maikhel.github.io/about#presentations