

### Ruby - Strings

```
"a string"      'a string'

"Concat" +     "#{concat}#"
"nation"      {nation}"

astring.length  astring.empty?

"a,b,c".split(' ', Result: [a,b,c]
')
```

Single quotes string, auto escape characters.

### Ruby - Array

```
foo =          Used for all examples
[a,b,c,d]      below

foo[n]         foo.first,
               foo.second

foo.length     foo.last == foo[-1]

foo.empty?     foo.include?(x)

foo.sort[!]    foo.shuffle[!]

foo.reverse[!  foo.push(x) or foo
]              << x

foo.join(",")  Result: "a, b, c, d"

foo[0..2]     Result: [a,b,c]

foo[1..-1]    Result: [b,c,d]

('a'..'d').to_a == foo
```

### Ruby - Block

```
3.times { puts "foo" }  Inline Block

(1..5).each do |i|     Multi Line Block
  puts 2 * i
end

%w[A B C].map { |char|
  char.downcase }

%w[A B C].map(&:downcase)
```

The last 2 lines result in an array of ["a", "b", "c"]. The %w creates an array of string. The last one is a weird Ruby shortcut.

### Ruby - Conditionals

```
puts "string is not empty" if
!astring.empty?

puts "string is not empty" unless
astring.empty?

if <expression>
  ...
else
  ...
end if
```

### Ruby - Hash + Symbol

```
foo = { "name" => "John", "id" =>
"JH" }

foo["name"]      ↗
                  "John"

foo = { name: "John", id: "JH" }

foo[:id]         ↗ "JH"

foo.each do |key, value| ↗ :name
  puts "#{key.inspect} - " - "John"
  "#{value.inspect}"     ↗ :id -
end                       "JH"
```

### Ruby - Objects

```
anobject.nil?    True or False

anobject.to_s    To String
```

### Ruby - Class

```
module BunchOfMethods
  def moduleMethod1
    ...
  end

  def moduleMethod2
    ...
  end
end

class MyClass < MySuperClass
  attr_accessor :attributeA,
:attributeB

  def initialize(attributes = {})

```

### Ruby - Class (cont)

```
@attributeA =
attributes[:attributeA]

@attributeB =
attributes[:attributeB]
end

def NiceString
  "#{@attributeA} #"
{@attributeA}"
end

end
```