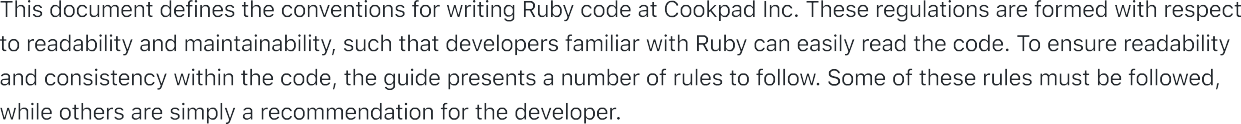


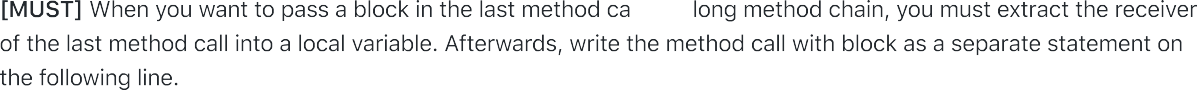
60eac9a

659 lines (514 sloc) 19.4 KB



%i

# 



# good

posts = Post.joins(:user). merge(User.paid). where(created\_at: target\_date)

posts.each do |post|

next if stuff\_ids.include?(post.user\_id) comment\_count += post.comments.size

end

# bad

posts = Post.joins(:user). merge(User.paid).

where(created\_at: target\_date).each do |post| next if stuff\_ids.include?(post.user\_id) comment\_count += post.comments.size

end



# good User.active.

some\_scope(foo). other\_scope(bar)

# bad User.active. some\_scope(foo). other\_scope(bar)









# coding: utf-8









 1\_000\_000.001\_023



0xABCD\_1234





 r



 1/2r #=> (1/2)

 Integer#quo   1.quo(2) #=> (1/2)

 i ri 

 1 + 2i #=> (1+2i)

 ''

 String.new





%



OPEN\_PARENTHESES = %!({[!

 Object#to\_s "#{obj.to\_s}"

\u \x



"\u{3333}" "\xE3\x8C\xB3"

\x 



 while until for



 String#+



 += String#<< String#concat









# good [ :foo,



x

(?# ... )



:bar,

:baz

(?: ... )

[

]

# bad [:foo,

:bar,

:baz

]

 =

# good array = [

:foo,

:bar,

:baz,

]

# bad

array = [ :foo,

:bar,

:baz, ]

# bad

array = [ :foo,

:bar,

:baz,

]

# bad array = [

:foo,

:bar,

:baz, ]

, 



 %w(...) %W(...) 

# good

words = %w(foo bar baz)

# bad

words = ['foo', 'bar', 'baz']

  [] 

 Array.new



 Array.new(n, obj) n  [obj] \* n



 [\*range]  Range#to\_a



# good

[\*1..10] #=> [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

# bad

(1..10).to\_a #=> [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

{  }



# good

{ hoge: 1, fuga: 2 } # bad

{hoge: 1, fuga: 2}

 { foo: 42 }



# good

{ first: 42, second: 'foo',

}

# bad

{ :first => 42,

:second => 'foo',

}

 { :foo => 42 }



:'foo.bar' :'foo-bar' if unless

# good

{ :cookpad => 42,

:'cookpad.com' => 'foo',

}

# bad

{ cookpad: 42,

:'cookpad.com' => 'foo',

}

  {} 

{



}

}

# good

{ first: 42, second: 'foo',

}

# bad

{

first: 42, second: 'foo',

}

# bad

{ first: 42, second: 'foo',

}

 = 



# good hash = {

first: 42, second: 'foo',

}

# bad

hash = { first: 42,

second: 'foo',

}

# bad

hash = { first: 42,

second: 'foo',

}

# bad

hash = { first: 42, second: 'foo',

}

# bad hash = { first: 42,

second: 'foo',

}

 , 



# good

{ foo: 1, bar: 2 }

# bad

{ 'foo' => 1, 'bar' => 2 }

{ key: value } 



:





 \*\*

 and or not



 or expression or raise 'message'



# good

fizzbuzz = if n % 3 == 0

n % 5 == 0 ? 'fizzbuzz' : 'fizz' else

n % 5 == 0 ? 'buzz' : "#{n}" end

# bad

fizzbuzz = n % 3 == 0 ? (n % 5 == 0 ? 'fizzbuzz' : 'fizz') : (n % 5 == 0 ? 'buzz' : "#{n}")

# bad

fizzbuzz = n % 3 == 0 ?

(n % 5 == 0 ? 'fizzbuzz' : 'fizz') :

(n % 5 == 0 ? 'buzz' : "#{n}")





## 



 unless condition if !condition



 until condition while !condition



 else  unless

 if unless case 



 then  :  if unless case



 while  until 



 do  :  while  until



||  unless  until





=



# good

result = if condition body\_code

end

# good result =

if condition body\_code

end

# bad

result = if condition body\_code

end

# bad

result = if condition

body\_code end









# good p(1 + 2)

# bad

p (1 + 2)

 { }



# good

foo(1, 2, foo: :bar, baz: 42)

# bad

foo(1, 2, { foo: :bar, baz: 42 })

  do end 

# 



{ }

# good

puts [1, 2, 3].map {|i| i \* i

}

# bad

puts [1, 2, 3].map do |i| i \* i

end

# good

[1, 2, 3].map {|n| n \* n

}.each {|n|

puts Math.sqrt(n)

}

# bad

[1, 2, 3].map do |n| n \* n

end.each do |n| puts Math.sqrt(n)

end



 do end 

# good

[1, 2, 3].each do |num| puts num

end

# bad

[1, 2, 3].each do |num| puts num

end

# bad

[1, 2, 3].each do |num|

puts num end

# bad

[1, 2, 3].each do |num| puts num end

 { 

{ } 



# good

[1, 2, 3].each {|num| puts num } [1, 2, 3].each { |num| puts num }

# bad

[1, 2, 3].each {|num| puts num}

# bad

[1, 2, 3].each { |num| puts num}

# good

10.times { puts 'Hello world' }

# bad

10.times {puts 'Hello world' }

# bad

10.times {puts 'Hello world'}

# bad

10.times { puts 'Hello world'}







(

)

Foo.new( arg,

long\_argument, key: value,

long\_key: long\_value, pretty\_so\_much\_very\_long\_key:

pretty\_so\_much\_very\_tooooooooooooooooooooo\_long\_value,

)



(

)

Foo.new(arg,

long\_argument, key: value,

long\_key: long\_value)



ActionMailer::Base.delivery\_method :smtp, host: 'localhost',

port: 25





 BEGIN  END 





  alias\_method alias 



 attr\_accessor attr\_reader attr\_writer  attr







self.

class << self

class Foo # good

def self.foo end

# bad

def Foo.foo end

end



class << self end

class Foo # good

class << self def foo

end

private :foo end

# bad

def self.foo end

class <<self private :foo

end end



private protected

public

class Foo # good def foo end

private :foo

# bad def foo end

private :foo end



private protected

public

# good class Foo

def foo end

private

def bar end

end

# bad class Foo

def foo end

private

def bar end

end

# bad class Foo

def foo end

private

end

def bar end

# bad class Foo

def foo end

private def bar

end end



# good

def your\_method(str)

new\_str = str.sub('xxx', 'yyy') end

# bad

def your\_method(str) str.sub!('xxx', 'yyy')

end

$foo

@@foo class\_attribute

i j