
RUBY
BITS
METHODS AND CLASSES
PRESS START

OPTIONAL ARGUMENTS

```ruby
def tweet(message, lat, long)
  #...
end

tweet("Practicing Ruby-Fu!", nil, nil)
```

location isn't always used, so let's add defaults

```ruby
def tweet(message, lat = nil, long = nil)
  #...
end

tweet("Practicing Ruby-Fu!")
```

location is now optional

RUBY BITS

NAMED ARGUMENTS - HASH

```ruby
def tweet(message, lat = nil, long = nil, reply_id = nil)
  #...
end
```

long parameter list

```ruby
tweet("Practicing Ruby-Fu!", 28.55, -81.33, 227946)
```

calls to it are hard to read

```ruby
tweet("Practicing Ruby-Fu!", nil, nil, 227946)
```

have to keep placeholders for arguments you're not using

RUBY BITS

HASH ARGUMENTS

```ruby
def tweet(message, options = {})
  status = Status.new
  status.lat = options[:lat]
  status.long = options[:long]
  status.body = message
  status.reply_id = options[:reply_id]
  status.post
end
```

hash argument

reference keys from hash

RUBY BITS

HASH ARGUMENTS

```ruby
def tweet(message, options = {})
```

```ruby
tweet("Practicing Ruby-Fu!",
  :lat => 28.55,
  :long => -81.33,
  :reply_id => 227946
)
```

keys show meaning

all combined into options argument

RUBY BITS

NAMED ARGUMENTS - HASH

Using Ruby 1.9 hash syntax

```
tweet("Practicing Ruby-Fu!",
    lat: 28.55,
    long: -81.33,
    reply_id: 227946
)
```

Repositioning the keys

```
tweet("Practicing Ruby-Fu!",
  reply_id: 227946,
  lat: 28.55,
  long: -81.33
)
```

RUBY
BITS

NAMED ARGUMENTS - HASH

Keys are optional

```
tweet("Practicing Ruby-Fu!",
  reply_id: 227946
)
```

Complete hash is optional

```
tweet("Practicing Ruby-Fu!")
```

RUBY BITS

EXCEPTIONS

```ruby
def get_tweets(list)
  if list.authorized?(@user)
    list.tweets
  else
    []
  end
end
```

"magic" return value

```ruby
tweets = get_tweets(my_list)
if tweets.empty?
  alert "No tweets were found!" +
    "Are you authorized to access this list?"
end
```

can't be sure it's an error

RUBY BITS

EXCEPTIONS

```ruby
def get_tweets(list)
  unless list.authorized?(@user)
    raise AuthorizationException.new
  end
  list.tweets
end
```

raise an Exception instead

```ruby
begin
  tweets = get_tweets(my_list)
rescue AuthorizationException
  warn "You are not authorized to access this list."
end
```

caller KNOWS there's a problem

RUBY BITS

"SPLAT" ARGUMENTS

```ruby
def mention(status, *names)
  tweet("#{names.join(' ')} #{status}")
end
```

RUBY BITS

YOU NEED A CLASS WHEN...

```
user_names = [
  ["Ashton", "Kutcher"],
  ["Wil", "Wheaton"],
  ["Madonna"]
]
user_names.each { |n| puts "#{n[1]}, #{n[0]}" }
```

OUTPUT:

```
Kutcher, Ashton
Wheaton, Wil
, Madonna
```

your users shouldn't have to deal with edge cases

RUBY BITS

YOU NEED A CLASS WHEN...

```ruby
class Name
  def initialize(first, last = nil)
    @first = first
    @last = last
  end
  def format
    [@last, @first].compact.join(', ')
  end
end
```

state!

behavior!

RUBY BITS

YOU NEED A CLASS WHEN...

```
user_names = []
user_names << Name.new('Ashton', 'Kutcher')
user_names << Name.new('Wil', 'Wheaton')
user_names << Name.new('Madonna')
user_names.each { |n| puts n.format }
```

OUTPUT:

```
Kutcher, Ashton
Wheaton, Wil
Madonna
```

◀ edge case handled!

RUBY BITS

OVERSHARING?

```ruby
class Tweet
  attr_accessor :status, :created_at
  def initialize(status)
    @status = status
    @created_at = Time.new
  end
end
```

```ruby
attr_accessor :baz
```

same as

```ruby
def baz=(value)
  @baz = value
end
def baz
  @baz
end
```

```ruby
tweet = Tweet.new("Eating breakfast.")
tweet.created_at = Time.new(2084, 1, 1, 0, 0, 0, "-07:00")
```

..... shouldn't be able to do this!

RUBY BITS

OVERSHARING?

```
class Tweet
  attr_accessor :status
  attr_reader :created_at
  def initialize(status)
    @status = status
    @created_at = Time.new
  end
end
```

doesn't define a setter!

```
tweet = Tweet.new("Eating breakfast.")
tweet.created_at =
  Time.new(2084, 1, 1, 0, 0, 0, "-07:00")
```

▶ undefined method 'created_at='

RUBY BITS

RE-OPENING CLASSES

```
tweet = Tweet.new("Eating lunch.")
puts tweet.to_s
```

```
...▶ #<Tweet:0x000001008c89e8>
```

not so readable...

```
class Tweet
  def to_s
    "#{@status}\n#{@created_at}"
  end
end
tweet = Tweet.new("Eating lunch.")
puts tweet.to_s
```

so just re-open the class and redefine it!

```
...▶ Eating lunch.
     2012-08-02 12:20:02 -0700
```

RUBY BITS

RE-OPENING CLASSES

- You can re-open and change any class.
- Beware! You don't know who relies on the old functionality.
- You should only re-open classes that you yourself own.

SELF

```ruby
class UserList
  attr_accessor :name
  def initialize(name)
    name = name
  end
end
```

this just re-sets the "name" local variable!

```ruby
list = UserList.new('celebrities')
list.name
```

nil

```ruby
class UserList
  attr_accessor :name
  def initialize(name)
    self.name = name
  end
end
```

but this calls "name=" on the current object

```ruby
list = UserList.new('celebrities')
list.name
```

"celebrities"

RUBY BITS


RUBY
BITS
METHODS AND CLASSES
PRESS START