Methods in Depth

Ruby Fundamentals

Overview

- Default parameter values
- Variable number of arguments
- Named arguments
- Aliasing
- Operators and operator overloading
- Method calls as messages
- method_missing and metaprogramming

Default Parameter Values

```ruby
def produce_spaceship(type = :freighter, size = :xl)
  # ...
end


def produce_spaceship(type = :freighter,
                      size = calc_default_size(type))
  # ...
end
```

- or -

```ruby
def produce_spaceship(type = :freighter,
                      size = (type == :freighter ? :xl : :m))
  # ...
end
```

Default Parameter Values

```
def produce_spaceship(type = :freighter, size = :xl,
                      engine_count)
  # ...
end

factory.produce_spaceship(4) # sets engine_count to 4
```

Variable Length Parameter Lists

```ruby
def produce_fleet(days_to_complete, *types)
  # ...
end

produce_fleet(10, :freighter, :freighter, :explorer)
```

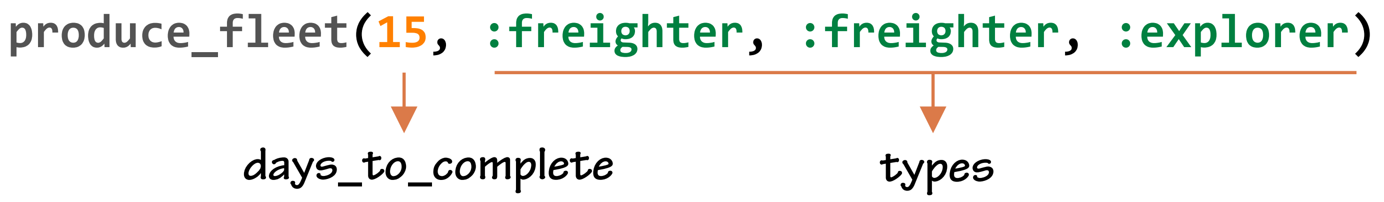
days_to_complete

types

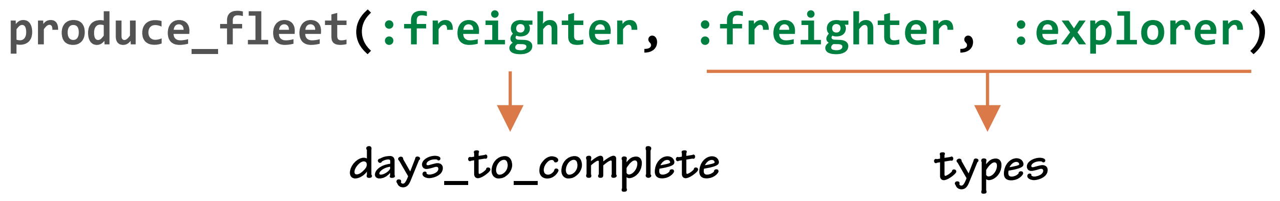
Variable Length Parameter Lists

```ruby
def produce_fleet(days_to_complete = 10, *types)
  # ...
end

produce_fleet
```



```ruby
produce_fleet(15, :freighter, :freighter, :explorer)
```



```ruby
produce_fleet(:freighter, :freighter, :explorer)
```

Variable Length Parameter Lists

```ruby
ship_types = [:freighter, :freighter, :explorer]
produce_fleet(15, *ship_types)
```

```ruby
produce_fleet(15, :freighter, :freighter, :explorer)
```

Keyword Arguments

```
produce_spaceship(:freigther, :m, 100, 4)


produce_spaceship(type: :freighter, size: :m,
                  fuel_tank_volume: 100, engine_count: 4)

def produce_spaceship(type: :freighter, size: :xl,
                      fuel_tank_volume: 400,
                      engine_count: 2)
  # ...
end
```

Keyword Arguments

```ruby
def produce_spaceship(type = :freighter,
                      size: :m, engine_count: 2)
  # ...
end
```

Keyword Arguments

```
def produce_spaceship(type = :freighter,
                      size: :m, **custom_components)

  components = {engine:    :standard,
                seats:     :standard,
                subwoofer: :none}

  components.merge!(custom_components)
  # ...
end

produce_spaceship(:yacht, size: :s,
                  engine: :rolls_royce, seats: :leather)
```

custom_components

Keyword Arguments

```ruby
build_params = {size: :s, engine: :rolls_royce,
                seats: :leather}

produce_spaceship(:yacht, build_params)

custom_components = {seats: :leather}
produce_spaceship(:yacht, size: :s, **custom_components)
```

Method Aliasing

```
alias_method
```

Operators

Logical operators

`&&` `||` `not` `and` `or` `?:`

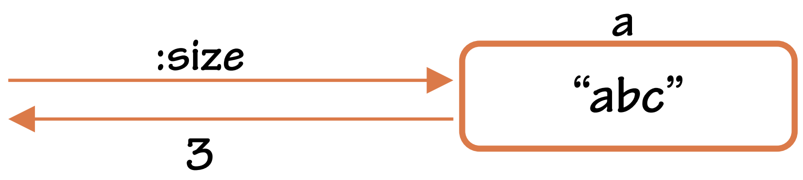
Assignment operators

`=` `+=` `-=` `*=` `/=` `%=` `**=` `&=` `|=` `^=`
`>>=` `<<=` `&&=` `||=`

Method Calls as Messages

```
a = "abc"
```

```
a.size
```



```
a.send(:size)
```

Method Calls as Messages

```
case input
when :up_arrow then ship.tilt_up
when :down_arrow then ship.tilt_down
when :left_arrow then ship.turn_left
when :right_arrow then ship.turn_right
end
```

```
handlers = {up_arrow: :tilt_up,
            down_arrow: :tilt_down,
            left_arrow: :turn_left,
            right_arrow: :turn_right}

ship.send(handlers[input])
```

```
ship.__send__(handlers[input])
```

Methods Outside Classes

```
double(10)
```

```
class Spaceship
  def launch
    batten_hatches
    # ...
  end
end
```

method_missing

`ship.xxyyzz` → NoMethodError

`method_missing`

method_missing

```
ship.cargo.find_by_destination("Earth")

ship.cargo.find_by_weight_and_volume(100, 10)
```

Other Metaprogramming Facilities

- *const_missing*
- Adding and removing methods at runtime
- *inherited* method

Summary

- Defining and calling modules
- Default parameters, keyword arguments, varargs
- Method aliasing
- Operators and operator overloading
- Method calls are messages
- method_missing