Gute Argumente

Argumente

Parameter & Argumente

- Klammern bei Definition und Aufruf optional, wenn nicht notwendig

```ruby
def method_name(arg1)
  # ...
end

method_name Time.new(2018, 10, 1)
```

Standardwerte

Parameter

- Optionaler Default-Wert
- Kann anschließend beim Aufruf weggelassen werden

```ruby
def evening(day = Time.new)
  Time.new day.year, day.month, day.day, 18
end

evening # => 2018-10-08 18:00:00 +0200
evening Time.new(2013, 5, 7)
# => 2013-05-07 18:00:00 +0200
```

Argumente

Argumentliste

- Beliebige Anzahl von Parametern akzeptieren

```ruby
def sum(a, *summands)
  summands.reduce(a) { |sum, num| sum + num }
end

sum
# => ArgumentError: wrong number of arguments (given 0, expected 1+)

sum 1        # => 1
sum 1, 5, 8 # => 14
```

Argumente

Liste zu Argumenten

- Methode mit Argumenten aus Liste aufrufen
- Übergibt jedes Element der Liste als jeweiliges Argument
- Anzahl der Elemente muss zur Anzahl der Parameter passen

```ruby
def add(a, b)
  a + b
end

nums = [1, 2]
add(*nums)

add(*[1, 2, 3])
# => ArgumentError: wrong number of arguments (given 3, expected 2)
```

Benannte Argumente

Keyword-Arguments

- Parameter mit Namen statt Position
- Hilfreich zum besseren Verständnis und bei vielen Parametern

```ruby
def final_pricing(subtotal, tax = 0.19, discount = 0)
  subtotal + (subtotal * tax) - discount
end

final_pricing 100              # => 119.0
final_pricing 100, 0.19, 10.0 # => 109.0
```

Benannte Argumente

Keyword-Arguments

- Optionale benannte Argumente mit Standardwert
- In beliebiger Reihenfolge angebbar

```ruby
def final_pricing(subtotal:, tax: 0.19, discount: 0)
  subtotal + (subtotal * tax) - discount
end

final_pricing subtotal: 100                  # => 119.0
final_pricing subtotal: 100, discount: 10.0  # => 109.0
final_pricing subtotal: 100, tax: 0.07       # => 107.0
```

Benannte Argumente

Keyword-Arguments

- Mit normalen Argumenten mischbar

```ruby
def final_pricing(subtotal, tax: 0.19, discount: 0)
  subtotal + (subtotal * tax) - discount
end

final_pricing 100                  # => 119.0
final_pricing 100, discount: 10.0  # => 109.0
final_pricing 100, tax: 0.07       # => 107.0
```

Benannte Argumente

Beliebige benannte Argumente

- Unbekannte Keyword-Arguments in Hash speichern

```ruby
def final_cart_pricing(items, discount: 0, **kwargs)
  subtotal = items.reduce(0) do |subtotal, item|
    discount += item.discount if item.discount?
    subtotal + item.price
  end

  final_pricing(subtotal, discount: discount, **kwargs)
end

# Cart: 2x 25€, 1x 50€ with 10€ discount
final_cart_pricing cart # => 109.0

# With extra loyalty discount
final_cart_pricing cart, discount: 10 # => 99.0

# With reduced tax
final_cart_pricing cart, tax: 0.07 # => 97.0
```

Block-Argument

Block-Argument speichern

- Statt `yield` den Block als Argument “einfangen”

```ruby
class Greeter
  def initialize(&format_block)
    @block = format_block
  end

  def format(message)
    @block.call message
  end
end

greeter = Greeter.new {|msg| "Hello, #{msg}!" }
greeter.format("John") # => "Hallo, John!"
greeter.format("Jane") # => "Hallo, Jane!"
```

Block-Argument

Hasso Plattner Institut

Variable als Block-Argument übergeben

- Gespeicherten Block als Block-Argument an Methode übergeben

```ruby
class Collector
  def initialize(&filter)
    @filter = filter
  end

  def process(list)
    list.select(&@filter)
  end
end

collector = Collector.new {|i| i.odd? }
collector.process(1..10)
# => [1, 3, 5, 7, 9]
```

Gute Argumente