Ruby Blocks

What are blocks?

- essentially anonymous functions
- temporarily switches context
- declared with either of two ways

```
{ puts “I am a block” }
```

```
do
  puts “I am a block”
end
```

Contrived Example

```
def great_function
  puts 'the Great Work has begun'
  outsource_labor 'first subtask'
  outsource_labor 'second subtask'
  puts 'the Great Work is completed'
end

def outsource_labor(subtask_name)
  puts “performing #{subtask_name}!”
end

great_function()

# => the Great Work has begun
# => performing first subtask!
# => performing second subtask!
# => the Great Work is completed
```

Contrived Example

```
def great_function
  puts 'the Great Work has begun'
  outsource_labor 'first subtask'
  outsource_labor 'second subtask'
  puts 'the Great Work is completed'
end

def outsource_labor(subtask_name)
  puts “performing #{subtask_name}!”
end

great_function()

# => the Great Work has begun
# => performing first subtask!
# => performing second subtask!
# => the Great Work is completed
```

```
def great_function
  puts 'the Great Work has begun'
  yield 'first subtask'
  yield 'second subtask'
  puts 'the Great Work is completed'
end

great_function do |subtask_name|
  puts "performing #{subtask_name}!"
end

# => the Great Work has begun
# => performing first subtask!
# => performing second subtask!
# => the Great Work is completed
```

Array#map

```
arr = [3,2,5]
arr.map! {|num| num*2}
puts arr

# => [6, 4, 10]
```

Custom map implementation

```
def custom_map!(arr)
  arr.each_with_index do |i, idx|
    arr[idx] = yield i
  end
end


my_arr = [3,2,5]
custom_map!(my_arr){ |val| val * 2 }
puts my_arr

# => [6, 4, 10]
```

Custom map implementation

```
def custom_map!(arr)
  arr.each_with_index do |i, idx|
    arr[idx] = yield i
  end
end


my_arr = [3,2,5]
custom_map!(my_arr){ |val| val * 2 }
puts my_arr

# => [6, 4, 10]
```

```
class Array
  def custom_map!
    self.each_with_index do |i, idx|
      self[idx] = yield i
    end
  end
end

my_arr = [3,2,5]
my_arr.custom_map! { |val| val * 2 }
puts my_arr

# => [6, 4, 10]
```

Array#select

```
things = [
  { id: 1, awesome: true  },
  { id: 2, awesome: false },
  { id: 3, awesome: true  }
]

things.select{|thing| thing[:awesome] == true }.count
# 2
```

Custom select implementation

```
def custom_select(source_arr)
  new_arr = []
  source_arr.each_with_index do |i, idx|
    new_arr << i if yield i
  end
  new_arr
end



things = [ { id: 1, awesome: true  },
           { id: 2, awesome: false },
           { id: 3, awesome: true  } ]


custom_select(things) {|thing| thing[:awesome] == true
}.count
# 2
```

Custom select implementation

```
def custom_select(source_arr)
  new_arr = []
  source_arr.each_with_index do |i, idx|
    new_arr << i if yield i
  end
  new_arr
end


things = [ { id: 1, awesome: true  },
           { id: 2, awesome: false },
           { id: 3, awesome: true  } ]


custom_select(things) {|thing| thing[:awesome] == true
}.count
# 2
```

```
class Array
  def custom_select(&block)
    new_arr = []
    self.each_with_index do |i, idx|
      new_arr << i if block.call(i) == true
    end
    new_arr
  end
end


things = [ { id: 1, awesome: true  },
           { id: 2, awesome: false },
           { id: 3, awesome: true  } ]


things.custom_select{|thing| thing[:awesome] == true }.
count
# 2
```

Working with ActiveRecord

```
orders = User.find(1337).orders

fedex_orders = orders.select{ |o| o.shipping_carrier == 'FedEx'  }
first_result = fedex_orders.map{ |o| o.id }
```

Working with ActiveRecord

```
orders = User.find(1337).orders

fedex_orders = orders.select{ |o| o.shipping_carrier == 'FedEx'  }
first_result = fedex_orders.map{ |o| o.id }

second_result = orders.map do |o|
  if o.shipping_carrier == 'FedEx'
    o.id
  else
    nil
  end
end.compact
```

Working with ActiveRecord

```
orders = User.find(1337).orders

fedex_orders = orders.select{ |o| o.shipping_carrier == 'FedEx'  }
first_result = fedex_orders.map{ |o| o.id }

second_result = orders.map do |o|
  if o.shipping_carrier == 'FedEx'
    o.id
  else
    nil
  end
end.compact

third_result = orders.map{ |o| o.shipping_carrier == 'FedEx' ? o.id : nil }.compact

first_result == second_result && second_result == third_result
# => true
```

Real-world example

https://github.com/honest/www/blob/line_item_adjustments/app/decorators/audit_logger/iterators/line_item_iterators.rb

```
def line_items_for_adjustment
  line_items.each do |li|
    next if li.is_in_order_insert? || li.is_kit_item? || li.is_bundle?
    yield li
  end
end



line_items_for_adjustment do |line_item|
  # do cool things with this eligible-for-adjustments line item
end
```

Qs?