Ruby: Collections

Array / Hash / Enumerable

Each

Runs a block of code for each element of a collection

Each

```
["carrots","apples","oranges"].each do |item|
  puts item
end
```

Each

```ruby
{name: "John", surname: "Doe"}.each do |key,value|
  puts "#{key} -> #{value}"
end
```

Exercise: Each

Create an array that has the names of 3 of your classmates. Use what you just learned to tell them good morning

```
classmates = ["Jack", "Wally", "Joey"]
```

Solution: Each

```ruby
classmates = ["Jack", "Wally", "Joey"]

classmates.each do |mate|
  puts "Good Morning, " + mate
end
```

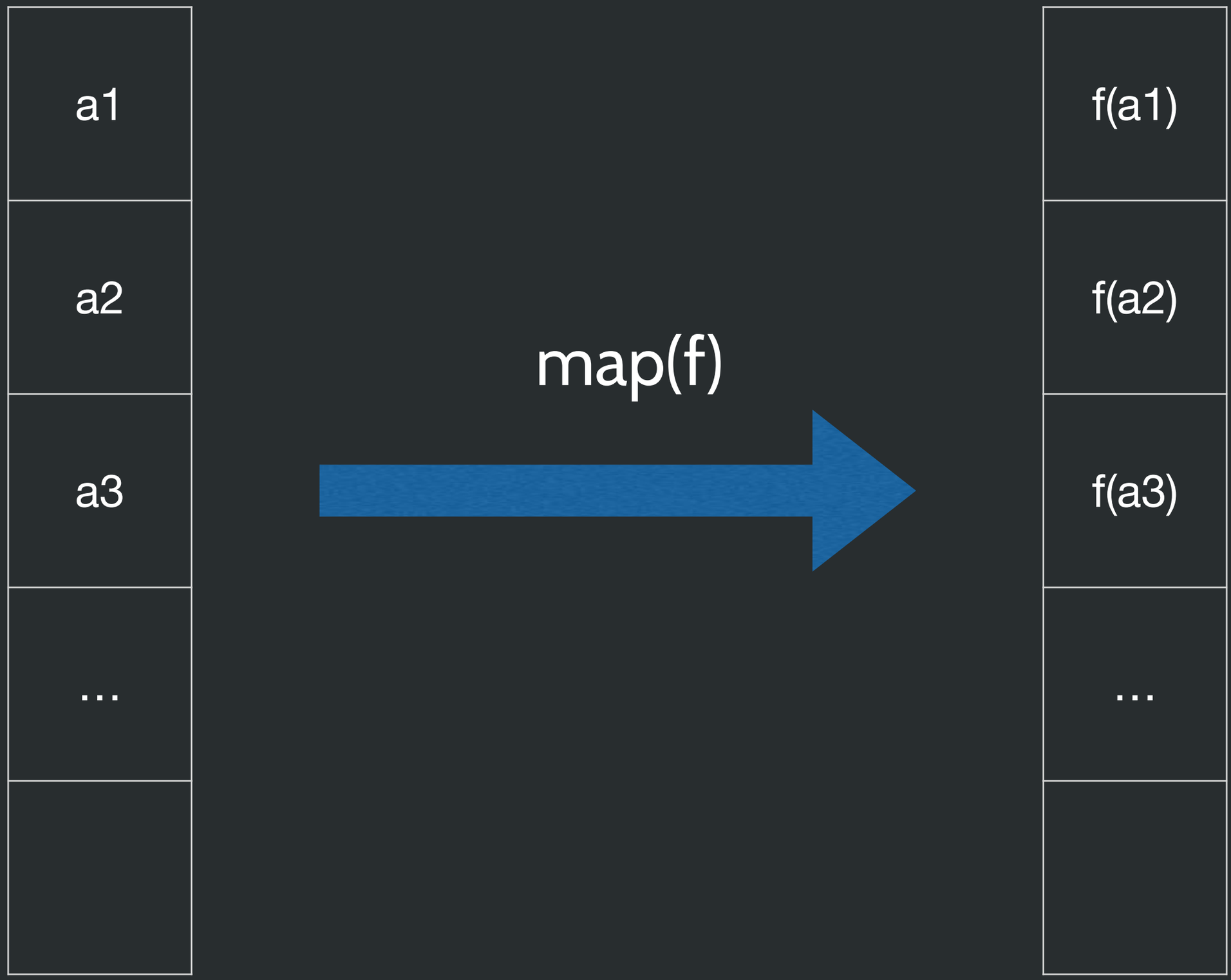
Map

Returns an array that is the result of applying a block of code to each of the elements of that collection

Map

```ruby
total = []
[1,2,3].each do |item|
  total.push(item+1)
end
total
```

```ruby
total = [1,2,3].map do |item|
  item+1
end
```


a1
a2
a3
...
map(f)
f(a1)
f(a2)
f(a3)
...

Exercise: Map

Given an Array with city names all in downcase, return another with those city names properly capitalized.

```
cities = ["miami", "madrid", "barcelona"]
```

Solution

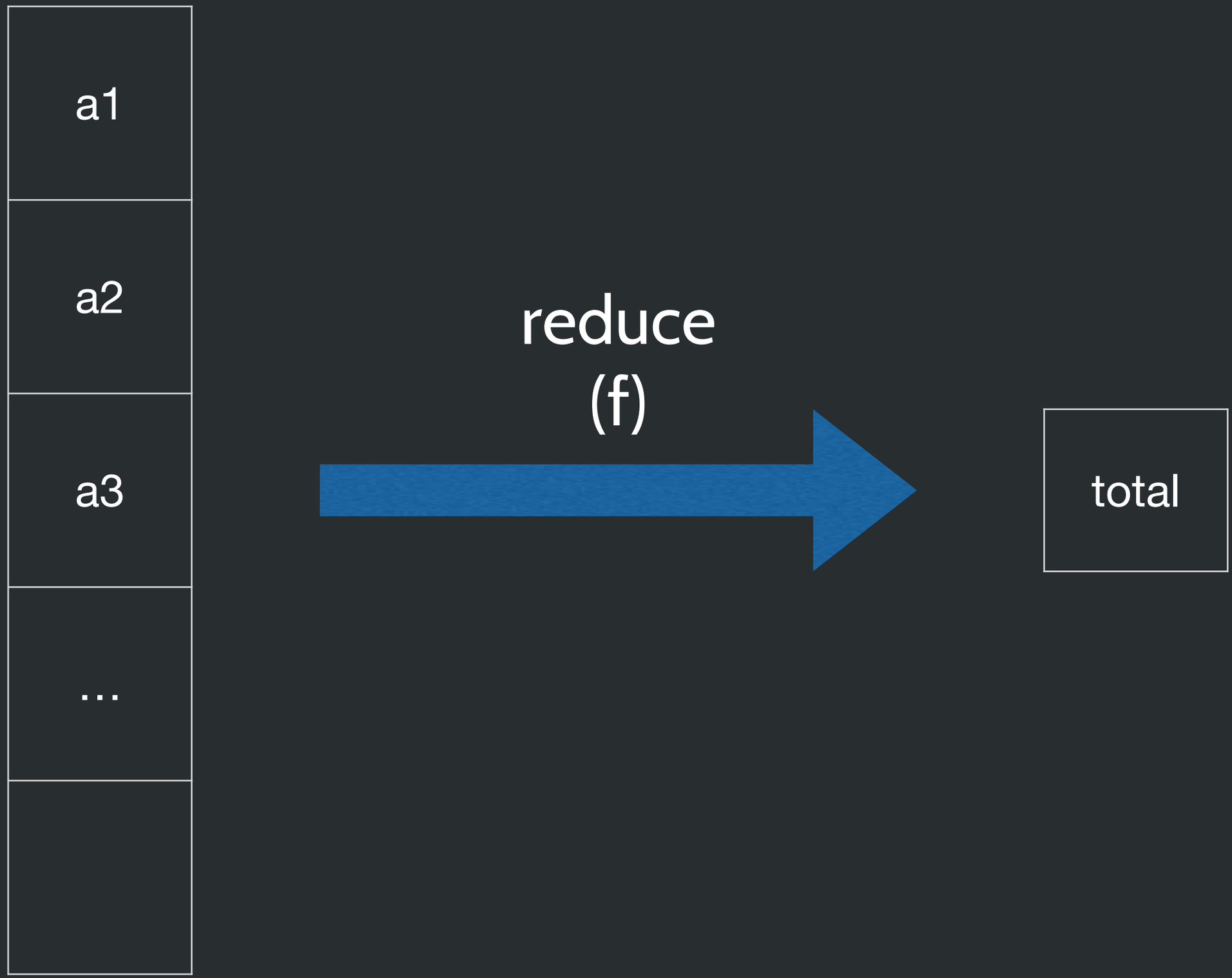
```
pretty_cities = cities.map {|city| city.capitalize}
```

Reduce

Transforms a collection into a single value applying a block of code many times

Reduce

```
[1,2,3].reduce{|sum,x| sum + x}
```


a1
a2
a3
...
reduce
(f)
total

Reduce

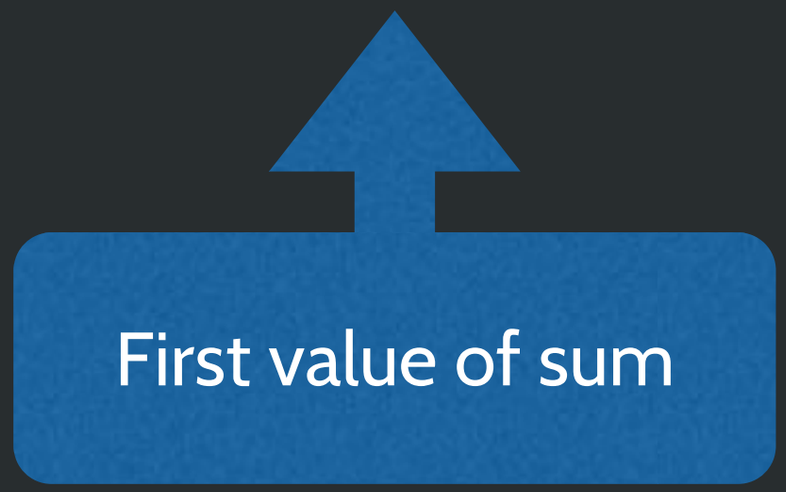
```
[1].reduce{|sum,x| sum + x}
```

Reduce

```
[].reduce{|sum,x| sum + x}
```

Reduce

```
[1,2,3].reduce{|sum,x| sum + x}
```



Reduce

```
[1].reduce{|sum,x| sum + x}
```

No first value of sum

Reduce

```
[].reduce{|sum,x| sum + x}
```

Returns nil

Reduce

```
[1,2,3].reduce(0){|sum,x| sum + x}
```

First value of sum

Reduce

```
[].reduce(0){|sum,x| sum + x}
```

Returns 0

Exercise: Reduce

Given an Array with city names, return the longest.

```
cities = ["miami", "madrid", "barcelona"]
```

Solution

```
cities.reduce do  |first, city|
   if first.length > city.length
      first
   else
      city
   end
end
```

Other collection methods

Each with index

```ruby
i = 0
["a","b","c"].each do |x|
  puts "[#{i}] #{x}"
  i += 1
end
```

```ruby
["a","b","c"].each_with_index {|x,i| puts "[#{i}] #{x}"}
```

Select

```ruby
[5,2,3].select {|x| x.odd?}
```

Grep

```
["casa", "masa", "pepino"].grep(/as/)
```

Find

```ruby
[5,2,3].find {|x| x.odd?}
```

Sort

```
[1,3,2].sort
```

Sort

```
[movie1, movie2, movie3].sort_by {|m| m.title}
```

The Enumerable Module

- The Enumerable module provides all the useful collection methods.
- All collections include the Enumerable module.
- The `each` method can be used for all of the above listed methods, but may not be the most efficient.