Filtrando con .each

Objetivo:

- Filtrar elementos de un arreglo.

Introducción

En muchos tipos de problemas necesitamos eliminar elementos de un arreglo en base a una condición. En este capítulo aprenderemos a hacerlo con el método `.each`

Filtrando con .each

Supongamos que tenemos un arreglo de notas y queremos mostrar todas las notas superiores a 5

```
array = [8, 2, 5.3, 7, 2, 9, 9, 6]
array.each do |ele|
  if ele > 5
    puts ele
  end
end
```

También sería posible generar un nuevo arreglo solo con esos elementos

```
array = [8, 2, 5.3, 7, 2, 9, 9, 6]
new_array = []
array.each do |ele|
  if ele > 5
    new_array.push(ele)
  end
end
```

Para ordenar mejor nuestro código podemos crear un método que filtre y devuelva un arreglo nuevo.

```
def filter(array, value)
    new_array = []
    array.each do |ele|
        if ele > value
            new_array.push(ele)
        end
    end
    new_array #devolvemos el arreglo nuevo
end

# Lo probamos
a = [8, 2, 5.3, 7, 2, 9, 9, 6]
filter(a, 5)
```

El retorno de .each

`.each` cuando termina de iterar devuelve el arreglo original, y Ruby devuelve implícitamente la última línea. Es por eso que tenemos el nombre del arreglo nuevo al final.

Más adelante aprenderemos que el método `.select` resuelve esto con mucho menos trabajos, pero es importante para nuestra formación profesional que desarrollemos las capacidades lógicas para resolver problemas con arrays.

¿Cómo saber cuándo ocupar .each o .times?

Si el problema no requiere los índices de los elementos del arreglo entonces podemos utilizar `.each` y resolver el problema con menos líneas de código, pero no hay problemas de ocupar `.times` si así lo prefieres.