Arreglos y hashes

Objetivos

- Convertir un hash en un arreglo.
- Convertir un arreglo en un hash.
- Transformar dos arreglos relacionados en un único arreglo.

Introducción

Los arreglos y los hashes tienen distintas ventajas, en algunos casos será más convenientes trabajar con hashes y en otras ocasiones será más sencillo trabajar con los arreglos.

Es importante aprender a diferenciar las situaciones ante las cuales nos enfrentamos para identificar la solución más simple.

Veamos un ejemplo

Supongamos que queremos almacenar alumnos y sus notas, perfectamente podríamos guardar la información en dos arrays, uno con los nombres y otro con las notas.

```
nombres = ['Alumno1', 'Alumno2', 'Alumno3']
notas = [10, 3, 8]
```

En este problema nos piden que nuestro programa haga dos cosas:

- Mostrar los alumnos con sus notas correspondientes.
- Mostrar la nota de un alumno dado su nombre.

Resolviendo el problema con arreglos

De esta forma si quisiéramos consultar la nota de un alumno podríamos recuperar la posición del alumno en el arreglo nombres y luego consultar el índice en notas.

```
index = nombres.index('Alumno1')
nota = notas[index]
```

Resolviendo el problema con hashes

Mientras que resolver el mismo problema con un hash puede ser mucho más sencillo.

```
products = {'product1' => 100,
            'product2' => 150,
            'product3' => 210}
puts products['product2']
```

Otro motivo importante

Algunos métodos reciben hashes y otros métodos reciben arreglos, tenemos que saber transformar los datos de un tipo a otro.

A lo largo del programa no siempre tendremos control sobre cuál es la estructura que estamos trabajando, por ejemplo puede ser que el resultado de un método devuelva un hash pero nosotros necesitemos un arreglo como entrada para otro método.

Convertir un hash en un arreglo

Es muy simple convertir un hash en un arreglo, simplemente tenemos que llamar al método `.to_a`.

```
{k1: 5, k2: 7}.to_a # => [[:k1, 5], [:k2, 7]]
```

Convirtiendo un arreglo en un hash

Podemos invertir la transformación ocupando el método `.to_h`

```
[[:k1, 5], [:k2, 7]].to_h
```

```
{:k1=>5, :k2=>7}
```

Cruzando información de dos arreglos

```
nombres = ['Alumno1', 'Alumno2', 'Alumno3']
notas = [10, 3, 8]
```

Si tenemos la información en dos arreglos y queremos guardarla en un hash tenemos dos opciones.

- Iterar el arreglo con un índice
- Utilizar el método `.zip`

Iterando el arreglo con índice

```
nombres = ['Alumno1', 'Alumno2', 'Alumno3']
notas = [10, 3, 8]

hash = {}
nombres.count.times do |i|
   element = nombres[i]
   nota = notas[i]
   hash[element] = nota
end

print hash
```

Iterando el arreglo elemento a elemento

Iterar el arreglo elemento a elemento también es posible, pero un poco mas complejo.

```
nombres = ['Alumno1', 'Alumno2', 'Alumno3']
notas = [10, 3, 8]

hash = {}
nombres.each do |nombre|
   i = nombres.index(nombre) # obtenemos el índice
   nota = notas[i]
   hash[nombre] = nota
end

print hash
```

Transformando con `.zip`

```
nombres = ['Alumno1', 'Alumno2', 'Alumno3']
notas = [10, 3, 8]
nombres.zip(notas).to_h # {"Alumno1"=>10, "Alumno2"=>3, "Alumno3"=>8}
```

Group by

`group_by` es un método de los arreglos que nos permite agrupar un conjunto de elementos bajo cualquier criterio, veamos algunos ejemplos. Este método entrega un hash con el resultado. Cada key del hash devuelto corresponde a un grupo y los valores son arrays con los elementos pertenecientes a cada grupo.

```
[1, 2, 3, 4, 5, 6, 7, 8] .group_by{ |x| x.even? }
```

```
[1, 2, 3, 4, 5, 6, 7, 8] .group_by{ |x| x > 4}
```

```
["hola", "a", "todos"].group_by { |x| x.length  } # Agrupar por el largo
```

```
[1, 2, 3, 4, 1, 2, 1, 5, 8].group_by { |x| x } # Agrupar por el mismo elemento
```

```
['a', 'ab', 'abc', 'b', 'bc', 'bcd'].group_by { |x| x[0] }  # Por la primera letra
```

```
['a', 1, 'b', 2, 'c'].group_by { |x| x.class } # Por tipo
```

`.group_by` es un método muy flexible y sirve para resolver diversos tipos de problemas.

Contando elementos dentro de un hash

Un uso relativamente frecuente de un hash es para contar elementos. Por ejemplo supongamos que tenemos un array con elementos y queremos contar la cantidad de veces que aparece cada uno de los elementos.

```
[1, 2, 6, 7, 2, 5, 8, 9, 1, 3, 6, 7]
```

Resolveremos el problema con dos estrategias distintas:

- Iterar y contar
- Agrupar y contar

Estrategia 1: Iterar y contar

Para resolver el problema iteraremos el arreglo, cada vez que encontremos un elemento nuevo lo guardaremos en un hash con la cuenta en uno, y si encontramos un elemento que ya está dentro del hash lo incrementamos en uno.

```
array = [1, 2, 6, 7, 2, 5, 8, 9, 1, 3, 6, 7]
hash = {}
array.each do |i|
  if hash[i]
    hash[i] += 1
  else
    hash[i] = 1
  end
end
puts hash
```

Estrategia 2 `.group_by`

```
# Los agrupamos
grouped = [1, 2, 6, 7, 2, 5, 8, 9, 1, 3, 6, 7] .group_by {|x| x}
# => {1=>[1, 1], 2=>[2, 2], 6=>[6, 6], 7=>[7, 7], 5=>[5], 8=>[8], 9=>[9], 3=>[3]}

#Luego remplazamos el valor por la cuenta
grouped.each do |k,v|
  grouped[k] = v.count
end
```