Formas de declarar un Hash

```ruby
person = { :name => "john doe", :age => 42, :height => 1.58 }
```

```ruby
person = { name: "john doe", age: 42, height: 1.58 }
```

```ruby
person = Hash[:name => "john doe", :age => 42, :height => 1.82 ]
```

TO_A => CONVERTIR A UN HASH EN UN ARRAY

```ruby
person = { :name => "john doe", :age => 42, :height => 1.58 }
puts person.to_a
# [[:name, "john doe"], [:age, 42], [:height, 1.58]]
```

El método .to_a retorna un nuevo Array con objetos Array con dos elementos el primero es la clave y el segundo el valor,

HAS_VALUE?(VALUE) => RETORNA TRUE SI EL VALOR EXISTE EN EL HASH

```ruby
person = { :name => "john doe", :age => 42, :height => 1.58 }
p person.has_value?(42) # true
```

El método .has_value?(value) retorna un booleano si el valor existe entre los valores del hash.

Formas de añadir entradas a un Hash

```ruby
colors = { "blue": "#00f","red": "#f00","green": "#0f0" }
colors.store(:yellow, "#ff0")
puts colors
#=> {:blue=>"#00f", :red=>"#f00", :green=>"#0f0", :yellow=>"#ff0"}
```

El método .store(key, value) recibe dos argumentos , la clave y el valor retornando el valor dado. Si la calve existe entonces modifica el valor de esa entrada.

REMOVER ENTRADAS DE UN HASH

```ruby
h = { :foo=>5, :bar =>3 }
h.delete(:foo)
puts  h
```

El método .delete remuve esa entrada usando su clave

```ruby
h = { :foo=>5, :bar =>3 }
h.clear
puts  h #=> {}
```

El método .clear remueve todos las entradas del hash.

EACH | EACH_PAIR

```
colores = { "azul": "#00f","rojo": "#f00","verde": "#0f0" }

colores.each_pair { |key, value| puts "#{key}: #{value}" }
```

El método .each_pair permite recorrer las entradas por cada clave-valor usando un bloque.

```
colores = { "azul": "#00f","rojo": "#f00","verde": "#0f0" }

colores.each { |key, value| puts "#{key}: #{value}" }
```

El método .each es un alias para .each_pair.