Symbols

Symbols

- First of all... How do they look like?

```
1.    :testing
```

They are like strings but without comillas and two periods before it.

Symbols

- A symbol is sometimes confused with variables, but it is not a variable
- A symbol is also not a string, although a string can also be declared into a symbol

```
[irb(main):001:0> :'symbol'
=> :symbol
```

Symbols

- Strings are used to work with data
- Just like strings represent data, symbols are these objects in Ruby which represent method & instance variable names.

```
1.    attr_reader :title
```

Symbols are identifiers
Identifier
  ● Variable names and methods names
It is the reason why we use them in attr_reader and other stuff.
Used for **meta-programming**

example
"".send()
best practice is to use a symbol for send (it asl accepts symbol)

```
3.0.0 :002 >
3.0.0 :003 >
3.0.0 :004 > "a long string".send(:to_sym)
 => :"a long string"
3.0.0 :005 > "a long string".to_sym
 => :"a long string"
3.0.0 :006 >
```
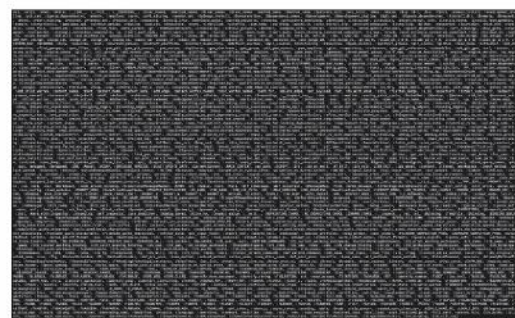
```
 => :"a long string"
3.0.0 :006 > "a long string".send(:gsub, 'string', 'hello')
 => "a long hello"
3.0.0 :007 >
```

Symbol.all_symbols

```
 => 7244
3.0.0 :003 > (:aa..:bb)
 => :aa..:bb
3.0.0 :004 > (:aa..:bb).map(&:to_sym)
 => [:aa, :ab, :ac, :ad, :ae, :af, :ag, :ah, :ai, :aj, :ak, :al, :am, :an, :ao, :ap, :aq, :ar, :as, :at, :au, :av, :aw, :ax, :ay, :az, :ba, :bb]
3.0.0 :005 >
```

Symbols - Hold on...

• What's the all_symbols table?
  • All symbols created until now in your program
  • Symbol.all_symbols

Symbols – Another common usage

- Metaprogramming as a best practice
  - Since send also accepts strings

```
1.    [1,2,3].send(:first)
```

Symbol to string
:hola.to_s

A symbol cannot be created twice. On the memory space it will be the same.
They point to the same memory space.

A more efficient way to create an array of symbols. You dont have to instantiate them with ":"

Symbols – An array of symbols

- Click to add text

```
1.    symbols = %i(a b c)
2.
3.    [:a, :b, :c]
```

Symbols - Remember

- You can use numbers in your symbols if they are not the first character
- Strings can be converted to symbols with the .to_sym method
- Symbols are immutable, thus they are preferred over strings
- When should you use them?
  - "If the textual content of the object is important, use a String, if the identity of the object is important, use a Symbol"

Hashes

- Collections of key-value pairs
- We have two options to define them

```
[irb(main):001:0> { :one => 1, :two => 2 }
=> {:one=>1, :two=>2}
irb(main):002:0> 
```

```
[irb(main):002:0> { one: 1, two: 2 }
=> {:one=>1, :two=>2}
irb(main):003:0> 
```

The second one is prefered for best practices

Map and each

Map returns an array
Each returns an iterator…

Hashes – Common methods

- hash == other_hash
  - Based on the same key-value pairs
- hash[key]
- hash.clear
  - Removes all key-value pairs from it
- .delete(key)

- .each_key { |key| block_goes_here }
- .has_value?(value)
- .inspect
  - Pretty print
- .shift
  - Removes the first key-value pair from the hash, returns it into a two-element array and removes it from the hash
- .to_s
  - Converts the hash into a string with the { } brackets

https://ruby-doc.org/core-3.0.0/Symbol.html
https://ruby-doc.org/core-3.0.0/Hash.html

We should be familiar with a couple of methods to be proficient with our programming.
Hay lugares donde los símbolos son preferibles y solo se pueden usar ahí.

Simbolos e strings
Se diferencian tambien en el object id.
Dos strings "hola" y "hola" son objetos diferentes. :hola y :hola son el mismo objeto. Nos sirven tambien para no crear mas espacio en la memoria.

Tarea de hoy
Crear o definir si los símbolos de un hash cuentan para el all symbols table

ruby paso a compilador just in time, por lo que si lo corremos fuera el irb entonces nos saldran otros valores

Se puede usar un mismo símbolo para múltiples cosas.

compilador just in time
Traducen codigo a codigo nativo en tiempo de ejecucion.

https://www.rubyguides.com/2018/02/ruby-symbols/