Ruby Symbols

One of Ruby’s unique features

Coder Factory Academy

What is a symbol?

- A symbol is for uniquely identifying something
- To make a symbol, you write a colon in front
- Symbols can be efficiently compared to each other
- You can still convert them to Strings

```
:first_name
:last_name

some_symbol = :first_name
:first_name == some_symbol
# => true

some_symbol.to_s
# => "first_name"
```

Symbols in Hashes

- Because symbols are unique, they are a perfect fit for Hashes as the keys
- You can either use a hash rocket **=>**
- Or, you can place the colon at the end of the key

```
hash = {
  :first_name => 'Alice',
  :last_name => 'Jones'
}

hash = {
  first_name: 'Alice',
  last_name: 'Jones'
}
```

Options to Methods

- This shorthand sugar syntax is also used in passing options to methods
- All methods support being called without parentheses: just use a space before the first argument
- If a Hash is the first argument, you can even drop the curly braces

```
def pass_me_options(options)
  first_name = options[:first_name]
  last_name = options[:last_name]
end

pass_me_options({ first_name:
'Alice', last_name: 'Jones' })

# Short cut
pass_me_options first_name:
'Alice', last_name: 'Jones'
```

Accepting options

- There’s also conveniences with accepting a Hash in a method
- In Ruby 2, you can list the keys as parameters, just use a trailing colon
- You can fallback to a default value for any key — here ‘Smith’ for **last_name**

```ruby
def pass_me_options(options)
  first_name = options[:first_name]
  last_name = options[:last_name]
  # Do something with first_name,
last_name
end

def pass_me_options(first_name:,
last_name: 'Smith')
  # Do something with first_name,
last_name
end
```