## In this lecture, we will discuss…

- ✧ Hashes
- ✧ **How** they are used
- ✧ **Why** they are used
- ✧ **Similarity** to blocks

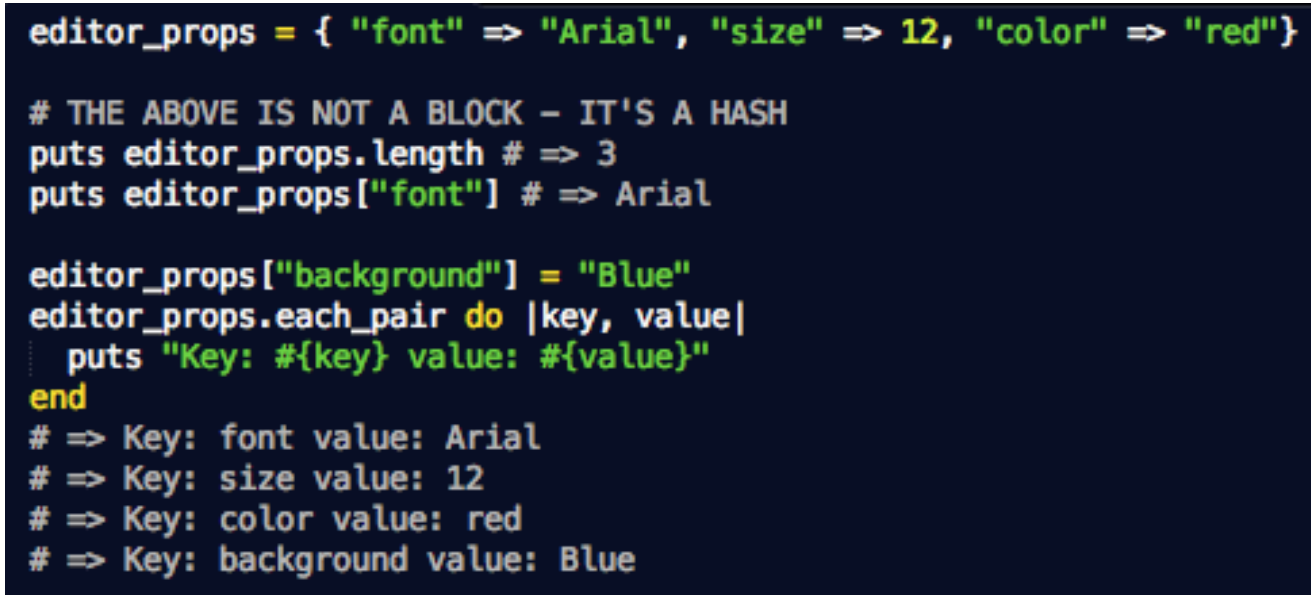
# Hashes

- ✧ **Indexed collections** of object references
- ✧ Created with either `{}` or `Hash.new`
- ✧ Also known as **associative arrays**
- ✧ Index(key) can be **anything**
  - **Not just an integer** as in the case of arrays

## Hashes

- Accessed using the `[]` operator
- Values set using
  - `=>` (creation)
  - `[]` (post creation)

# Hashes



```ruby
editor_props = { "font" => "Arial", "size" => 12, "color" => "red"}

# THE ABOVE IS NOT A BLOCK - IT'S A HASH
puts editor_props.length # => 3
puts editor_props["font"] # => Arial

editor_props["background"] = "Blue"
editor_props.each_pair do |key, value|
  puts "Key: #{key} value: #{value}"
end
# => Key: font value: Arial
# => Key: size value: 12
# => Key: color value: red
# => Key: background value: Blue
```

# Hashes

✧ What if you try to **access a value** in the Hash for which an entry **does not exist?**

- `nil` is returned

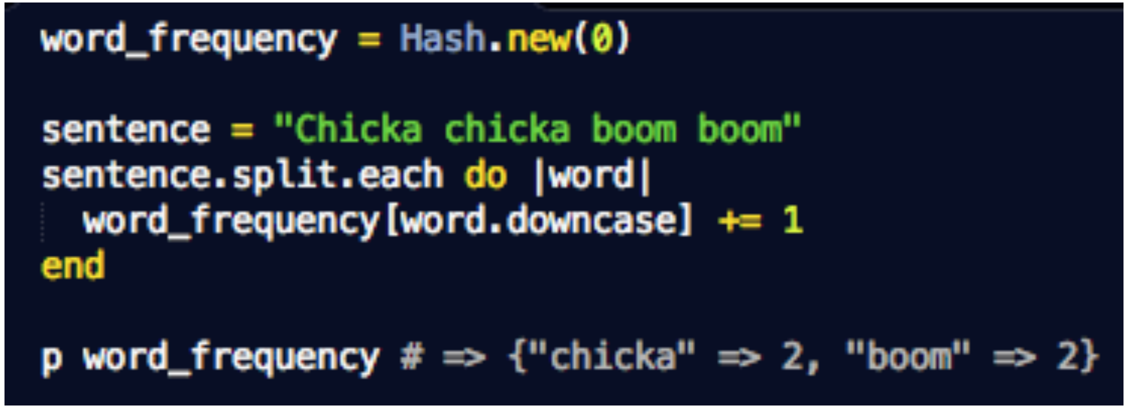
✧ If a Hash is created with `Hash.new(0)` ← 0 is just an example

`0` is returned instead

**Hashes API is also very important to master!**

http://ruby-doc.org/core-2.2.0/Hash.html

## Hashes



```
word_frequency = Hash.new(0)

sentence = "Chicka chicka boom boom"
sentence.split.each do |word|
  word_frequency[word.downcase] += 1
end

p word_frequency # => {"chicka" => 2, "boom" => 2}
```

# More Hashes

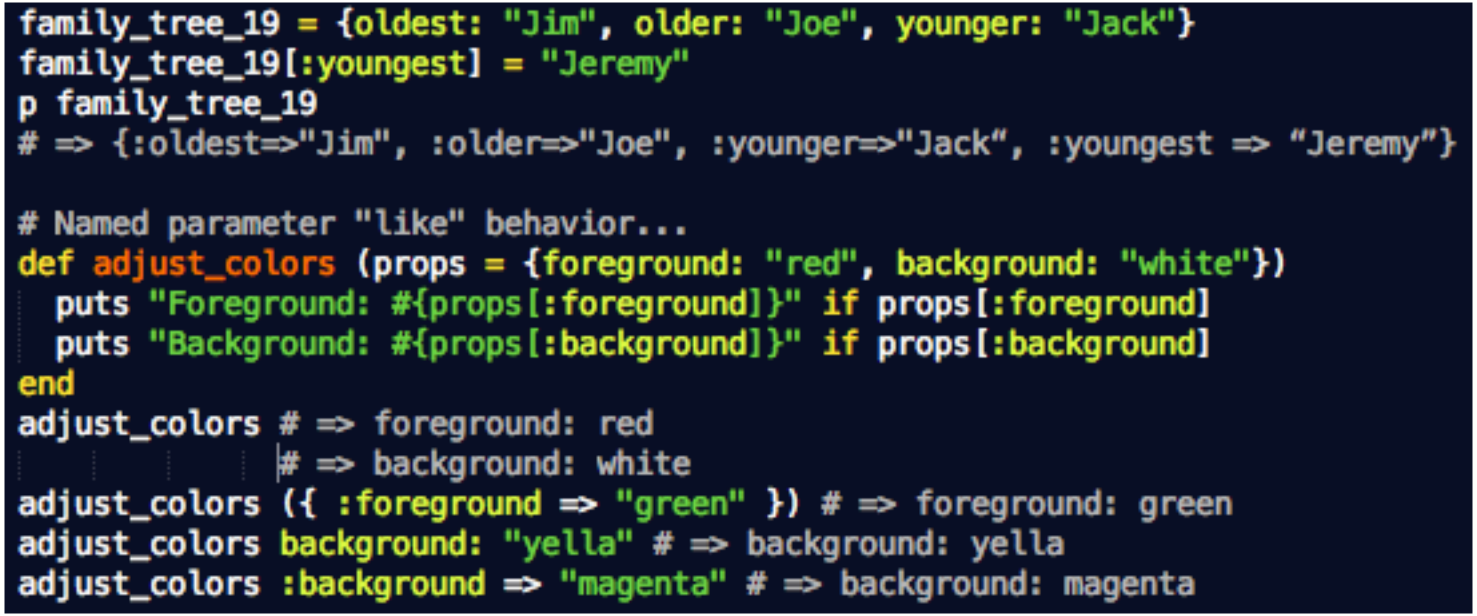
- As of **Ruby 1.9**
  - The order of putting things into `Hash` maintained
  - If using symbols as keys – can use `symbol:` syntax

# More Hashes

- If a `Hash` is the **last argument** to a method `{ }` are optional

**Last argument not including a block!**

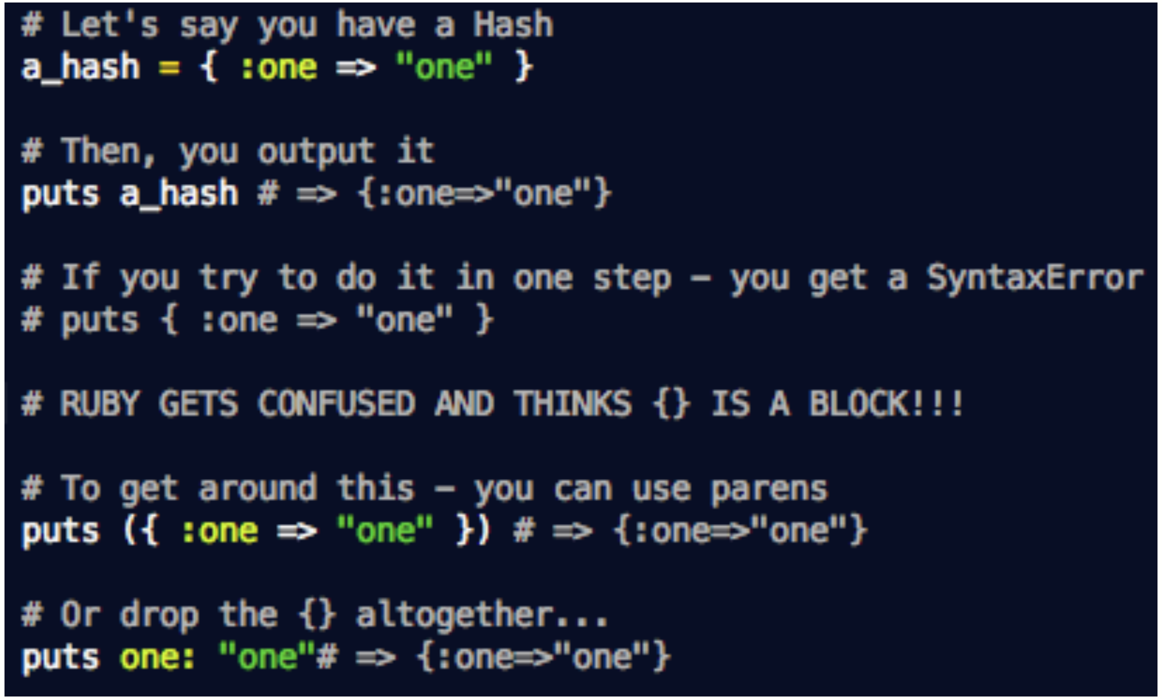
## Hashes



```ruby
family_tree_19 = {oldest: "Jim", older: "Joe", younger: "Jack"}
family_tree_19[:youngest] = "Jeremy"
p family_tree_19
# => {:oldest=>"Jim", :older=>"Joe", :younger=>"Jack", :youngest => "Jeremy"}

# Named parameter "like" behavior...
def adjust_colors (props = {foreground: "red", background: "white"})
  puts "Foreground: #{props[:foreground]}" if props[:foreground]
  puts "Background: #{props[:background]}" if props[:background]
end
adjust_colors # => foreground: red
              # => background: white
adjust_colors ({ :foreground => "green" }) # => foreground: green
adjust_colors background: "yella" # => background: yella
adjust_colors :background => "magenta" # => background: magenta
```

# Block and Hash Confusion



```ruby
# Let's say you have a Hash
a_hash = { :one => "one" }

# Then, you output it
puts a_hash # => {:one=>"one"}

# If you try to do it in one step - you get a SyntaxError
# puts { :one => "one" }

# RUBY GETS CONFUSED AND THINKS {} IS A BLOCK!!!

# To get around this - you can use parens
puts ({ :one => "one" }) # => {:one=>"one"}

# Or drop the {} altogether...
puts one: "one"# => {:one=>"one"}
```

# Summary

- Hashes are **indexed collections**
- Usage is **very similar to regular arrays**
- Although uncommon, can be **confused for blocks**

**What's next?**

- Classes