Nhớ lại rằng băm là [tập hợp](https://www.codecademy.com/courses/learn-ruby/lessons/ruby-data-structures/exercises/introduction-to-hashes) các cặp khóa-giá trị, trong đó một khóa duy nhất được liên kết với một số giá trị. Ví dụ:

breakfast = {   
  "bacon" => "tasty",  
  "eggs" => "tasty",  
  "oatmeal" => "healthy",  
  "OJ" => "juicy"  
}

Hãy nhớ rằng các khóa phải là duy nhất, nhưng các giá trị có thể lặp lại. Đó là lý do tại sao chúng ta có thể có nhiều hơn một khóa chia sẻ giá trị “tasty.”)

Chúng ta có thể tạo hàm băm theo một số cách, nhưng hai trong số những cách phổ biến nhất là

1. **ký hiệu chữ băm** :

new\_hash = { "one" => 1 }

và

2. **ký hiệu hàm tạo băm** :

new\_hash = Hash.new

**What's a Symbol?**

Bạn có thể coi biểu tượng Ruby như một loại tên. Điều quan trọng cần nhớ là các ký hiệu không phải là chuỗi:

"string" == :string # false

Trên và ngoài các cú pháp khác nhau, có một hành vi chính của các ký hiệu khiến chúng khác với chuỗi. Mặc dù có thể có nhiều chuỗi khác nhau đều có cùng giá trị, nhưng chỉ có một bản sao của bất kỳ ký hiệu cụ thể nào tại một thời điểm nhất định.

**Symbol Syntax**

Các ký hiệu luôn bắt đầu bằng dấu hai chấm ( :). Chúng phải là tên biến Ruby hợp lệ, vì vậy ký tự đầu tiên sau dấu hai chấm phải là một chữ cái hoặc dấu gạch dưới ( \_); sau đó, mọi kết hợp chữ cái, số và dấu gạch dưới đều được phép.

Đảm bảo rằng bạn không đặt bất kỳ khoảng trắng nào trong tên biểu tượng của mình — nếu bạn làm vậy, Ruby sẽ bị nhầm lẫn.

:my symbol # Don't do this!  
:my\_symbol # Do this instead.

**Biểu tượng được sử dụng để làm gì?**

Các biểu tượng xuất hiện ở rất nhiều nơi trong Ruby, nhưng chúng chủ yếu được sử dụng làm khóa băm hoặc để tham chiếu đến tên phương thức. (Chúng ta sẽ xem cách các biểu tượng có thể tham chiếu các phương thức trong bài học sau.)

sounds = {  
  :cat => "meow",  
  :dog => "woof",  
  :computer => 10010110,  
}

Các ký hiệu tạo ra các khóa băm tốt vì một số lý do:

1. Chúng là bất biến, có nghĩa là chúng không thể thay đổi sau khi chúng được tạo;
2. Chỉ một bản sao của bất kỳ ký hiệu nào tồn tại tại một thời điểm nhất định, vì vậy chúng tiết kiệm bộ nhớ;
3. Symbol-as-key nhanh hơn string-as-key vì hai lý do trên.

**Chuyển đổi giữa các biểu tượng và chuỗi**

Việc chuyển đổi giữa chuỗi và ký hiệu chỉ diễn ra trong tích tắc.

:sasquatch.to\_s  
# ==> "sasquatch"  
   
"sasquatch".to\_sym  
# ==> :sasquatch

Các .to\_svà .to\_symphương pháp là những gì bạn đang tìm kiếm!

Ngoài việc sử dụng .to\_sym, bạn cũng có thể sử dụng .intern. Điều này sẽ nội bộ chuỗi thành một biểu tượng và hoạt động giống như .to\_sym:

"hello".intern  
# ==> :hello

Khi bạn đang xem mã của người khác, bạn có thể thấy .to\_symhoặc .intern(hoặc cả hai!) Khi chuyển đổi chuỗi thành ký hiệu.

**The Hash Rocket Has Landed**

Tin tốt là cú pháp đã thay đổi dễ nhập hơn cú pháp tên lửa băm cũ và nếu bạn đã quen với các đối tượng JavaScript hoặc từ điển Python, nó sẽ trông rất quen thuộc:

new\_hash = {   
  one: 1,  
  two: 2,  
  three: 3  
}

Hai thay đổi là:

1. Bạn đặt dấu hai chấm ở cuối biểu tượng, không phải ở đầu;
2. Bạn không cần tên lửa băm nữa.

Điều quan trọng cần lưu ý là mặc dù các phím này có dấu hai chấm ở cuối thay vì ở đầu, chúng vẫn là biểu tượng!

puts new\_hash  
# => { :one => 1, :two => 2, :three => 3 }

Từ bây giờ, chúng tôi sẽ sử dụng cú pháp băm 1,9 khi đưa ra các ví dụ hoặc cung cấp mã mặc định. Bạn sẽ muốn làm quen với kiểu tên lửa băm khi đọc mã của người khác, có thể cũ hơn.

**so sánh**

Chúng tôi đã đề cập rằng tra cứu mã băm nhanh hơn với các phím biểu tượng so với các phím chuỗi. Đây, chúng tôi sẽ chứng minh điều đó!

Mã trong trình soạn thảo sử dụng một số cú pháp mới, vì vậy đừng lo lắng về việc hiểu tất cả. Nó xây dựng hai hàm băm bảng chữ cái: một hàm ghép các ký tự chuỗi với vị trí của chúng trong bảng chữ cái (“a” với 1, “b” với 2…) và một hàm sử dụng ký hiệu ( :avới 1, :bvới 2…). Chúng tôi sẽ tra cứu ký tự “r” 100.000 lần để xem quy trình nào chạy nhanh hơn!

Thật tốt khi lưu ý rằng các con số bạn sẽ thấy chỉ cách nhau một phần nhỏ của giây và chúng tôi đã thực hiện tra cứu băm *100.000 lần* mỗi số. Việc sử dụng các biểu tượng trong trường hợp này không làm tăng nhiều hiệu suất, nhưng chắc chắn là có!

require 'benchmark'

string\_AZ = Hash[("a".."z").to\_a.zip((1..26).to\_a)]

symbol\_AZ = Hash[(:a..:z).to\_a.zip((1..26).to\_a)]

string\_time = Benchmark.realtime do

  100\_000.times { string\_AZ["r"] }

end

symbol\_time = Benchmark.realtime do

  100\_000.times { symbol\_AZ[:r] }

end

puts "String time: #{string\_time} seconds."

puts "Symbol time: #{symbol\_time} seconds."

String time: 0.006987667999055702 seconds.

Symbol time: 0.004695768002420664 seconds.

**Trở nên chọn lọc hơn**

Chúng ta biết cách lấy một giá trị cụ thể từ một hàm băm bằng cách chỉ định khóa được liên kết, nhưng nếu chúng ta muốn lọc một hàm băm cho các giá trị đáp ứng các tiêu chí nhất định thì sao? Đối với điều đó, chúng ta có thể sử dụng .select.

grades = { alice: 100,  
  bob: 92,  
  chris: 95,  
  dave: 97  
}  
   
grades.select { |name, grade| grade <  97 }  
# ==> { :bob => 92, :chris => 95 }  
   
grades.select { |k, v| k == :alice }  
# ==> { :alice => 100 }

1. Trong ví dụ trên, đầu tiên chúng ta tạo một gradeshàm băm ánh xạ các ký hiệu thành các số nguyên.
2. Sau đó, chúng tôi gọi .selectphương thức và chuyển vào một khối mã. Khối chứa một biểu thức để chọn các cặp khóa / giá trị phù hợp. Nó trả về một hàm băm chứa :bobvà :chris.
3. Cuối cùng, chúng tôi gọi lại .selectphương thức. Khối của chúng tôi chỉ tìm kiếm chìa khóa :alice. Đây là một phương pháp không hiệu quả để lấy một cặp khóa / giá trị, nhưng nó cho thấy điều .selectđó không sửa đổi hàm băm.

**Nhiều phương pháp hơn, nhiều giải pháp hơn**

Công việc tuyệt vời!

Chúng tôi thường thấy rằng chúng tôi chỉ muốn khóa hoặc giá trị được liên kết với một cặp khóa / giá trị và thật khó khăn khi đặt cả hai vào khối của chúng tôi và chỉ hoạt động với một. Chúng ta có thể lặp lại *chỉ* các khóa hay *chỉ* các giá trị không?

Đây là Ruby. Tất nhiên là chúng ta có thể.

Ruby bao gồm hai phương thức băm .each\_keyvà .each\_valuethực hiện chính xác những gì bạn mong đợi:

my\_hash = { one: 1, two: 2, three: 3 }  
   
my\_hash.each\_key { |k| print k, " " }  
# ==> one two three  
   
my\_hash.each\_value { |v| print v, " " }  
# ==> 1 2 3