Working With Ruby Hashes

Working With Ruby Hashes

Published October 13, 2013
Last updated January 3, 2017
3 comments

This article will cover the basics of working with ruby hashes.

This article will cover the basics of working with hashes in Ruby. Hashes are also known as "dictionaries" or "associative arrays" in the programming world. They provide a useful means of storing data.

All examples in this article can be run using the `irb` console command. This will get you to a Ruby prompt. If you want to exit the Ruby prompt at any time, merely type `exit` .

Creating a New Hash

There are many different ways of creating a new hash. All yield the same result. The easiest way is listed below.

```
{:ruby => "hashes"}
```

However, you can also create the hash using the methods listed below.

```
Hash.[]("ruby", "hashes", "are", "awesome!")
Hash.[]("ruby" => "hashes", "are" => "awesome!")
Hash["ruby", "hashes", "are", "awesome!"]
Hash["ruby" => "hashes", "are" => "awesome!"]
{"ruby", "hashes", "are", "awesome!"}
```

Finally, you can use the new Ruby syntax of placing the colon (`:`) to the right instead of the left.

```
{hashes: "awesome", ruby: "awesome"}
```

Accessing Data in a Hash

It's easy to get data back from a hash. Simply use the code listed below.

```
h = {hashes: "awesome", ruby: "awesome"}
h[:hashes] # returns "awesome"
```

You can also use `fetch` .

```
h.fetch(:hashes) # returns awesome
```

Adding New Data to a Hash

Adding new data to a hash is simple.

```
h[:rails] = "rocks" # h is now {:hashes => "awesome", :ruby => "awesome", :rails => "rocks"}
```

You can also use this code to update an existing key/value pair.

Deleting Data From a Hash

You can easily delete data from a hash using the `delete` method.

```
h = {hashes: "awesome", ruby: "awesome", rails: "awesome"}
h.delete(:rails) # h is now {:hashes => "awesome", :ruby => "awesome"}
```

You can delete all key/value pairs using the clear method.

```
h.clear # h is now {}
```

You can also use `reject` and `reject!` to set conditions on key/value removal, reject creates a copy of the hash and reject! returns nil if nothing was removed.

```
h = {hashes: "awesome", ruby: "awesome", rails: "awesome"}
h.reject {|key, value| key == :rails} # returns a new hash containing {:hashes => "awesome", :ruby => "awesome"}, h is untouched.
h.reject! {|key, value| key == :rails} # h is now {:hashes => "awesome", :ruby => "awesome"}
```

Iterating Through a Hash

You can easily iterate through the items in a hash.

```
h = {hashes: "awesome", ruby: "awesome", rails: "awesome"}
h.each do |key, val|
  puts "#{key} => #{val}" # prints each key and value.
end
```

In addition, if you only need the keys you can use the following code.

```
h.each_key do |key|
  puts "#{key}" # prints each key.
end
```

Also, the values:

```
h.each_value do |val|
  puts "#{value}" # prints each value.
end
```

Detecting Keys and Values in a Hash

You can detect whether a key exists using any of the following methods.

```
h = {hashes: "awesome", ruby: "awesome", rails: "awesome"}
h.has_key?(:hashes) # true
h.include?(:hashes) # true
h.key?(:hashes) # true
h.member?(:hashes) # true
```

You can easily determine whether a hash has data in it or not using `empty?` .

```
h.empty? # false
```

In addition, you can see how many key/value pairs are in the hash using `length` :

```
h.length # 3
```

You can use `detect` and it's alias `find` to find values in a hash. These methods return the first resulting match:

```
h.detect {|key, val| val == "awesome"} # returns [:hashes, "awesome"]
h.find {|key, val| val == "awesome"} # returns [:hashes, "awesome"]
```

You can also return all matches using `select` or `find_all` :

```
h.select { |key, val| val == "awesome" } # returns {:hashes => "awesome", :ruby => "awesome", :rails => "awesome"}
h.find_all [[:hashes, "awesome"], [:ruby, "awesome"], [:rails, "awesome"]]
```

Sorting a Hash

You can do basic sorting on a hash:

```
h2 = {"b" => "3", "a" => "1", "c" => "5"}
h2.sort # returns [["a", "1"], ["b", "3"], ["c", "5"]]
```

Or you can sort in reverse:

```
h2.sort {|k1, k2| k2 <=> k1}
```

Merging Two hashes

You can easily merge multiple hashes using the `update` method.

```
h1 = {"a" => 1, "b" => 2}
h2 = {"c" => 3, "d" => 4}
h1.update(h2) # h1 is now {"a" => 1, "b" => 2, "c" => 3, "d" => 4}
```

Duplicate a Hash

You can easily duplicate a hash using the `dup` method.

```
h1 = {hashes: "awesome", ruby: "awesome", rails: "awesome"}
h2 = Marshal.load( Marshal.dump(h1))
h2.clear # h1 is still {:hashes => "awesome", :ruby > "awesome", :rails > "awesome"}
```

That's it! Thanks for reading!

Skipping Validations in Ruby On Rails

Rails Console Debugging Tricks

Easier Debugging With Pry

Getting Started with JBuilder

Rails 4 Preview: Strong Parameters

Code Concerns in Rails 4 Models

Basic AJAX in Ruby on Rails

Renaming a Database Column in Ruby on Rails

Comments

Note: You are not logged in. This means your comment be automatically sent into moderation and you won't receive emails when a user replies. To log in using GitHub, click the button below. This is a one step process.

Your information is kept private. to RichOnRails.com and will never be given away or sold. Take a look at our Privacy Policy (/privacy) for more details.

Sign In with Github (/auth/github)

Comment Anonymously

HTML is NOT allowed. However, you can use markdown. See Examples

Comment

Guest User
Posted 30 days ago

How do you print the first value in your hash?

Guest User
Posted 15 days ago

By using

```
{}.values.first
```

.
Example:
h={:a=>1, :b=>2}
h.values.first => 1

Reply

Guest User
Posted about 1 month ago

There is a error method is not dub but dump in:
You can easily duplicate a hash using the dup method.

Reply

Syncfusion
SQL Server for C# Developers Succinctly by Sander Rossel
Data Capture and Extraction with C# Succinctly by Ed Freitas
Customer Success for C# Developers Succinctly by Ed Freitas
C# Code Contracts Succinctly by Dirk Strauss
Object-Oriented Programming in C# Succinctly by Sander Rossel
DOWNLOAD
FREE E-BOOKS
100+ Titles
100 Pages
Kindle & PDF Formats

SOCIAL MEDIA

Facebook (https://facebook.com/richonrails)
Twitter (https://twitter.com/RichIsOnRails)
YouTube (https://www.youtube.com/channel/UCjSai7773YaHAGNNGx78O0Q)

READ

Articles (/articles)
Blog (/posts)

ABOUT US

About Us (/about)
Terms of Use (/legal)
Privacy Policy (/privacy)
Contact Us (/contact)
Advertise With Us (/contact?subject=advertising)

NEWSLETTER

Sign up for our weekly newsletter to receive news, Ruby tips, and more.

you@example.com

Subscribe

R RichOnRails (/)

f (https://facebook.com/richonrails) (https://twitter.com/richisonrails) (/?format=atom)