The Solution

This solution would be very easy. But roman numbers are tricky for some kinds of numbers, like: 4, 9. So I had to read the roman number in a reverse order, from right to the left. For each roman char, I looked for it in a list for a match and added it to a sum.

Tests

I used RSpec for the test suite. I tested all tricky scenarios that I would have to validate in the code.

I also used Coco gem for the code coverage, it's a 'new' gem. It displays the code coverage for projects running Ruby 2.0 or newer.

I done TDD during the code challenge and it helped me to take some decisions and make the code cleaner and readable.

Code style

I used rubocop gem as well. It's a static code analyzer for Ruby and it validates the code using the Ruby style guide. It's a very nice gem, you can validate it for some things in your code that you can miss.

I think it's a nice gem for team usage. So the code will be pretty similar. It also validates that you're running Ruby 1.9+ and validates your code to use 1.9+ features like the new hash syntax.

Misc

I used Git as repository to make a journal of my code. Every major code change, I commited. So with git log/git show you can follow the code evolution during the time.

How to run

Please, make sure you have Ruby 2.x version or newer installed and then:

```
$ gem install bundler
```

```
$ bundle install
```

And execute the Ruby script:

```
$ ruby roman_to_integer.rb
```