Beyond This Tutorial

Chapter 11

So where do we go now? If you have a question, who can you ask? What if you want your program to open a webpage, send an email, or resize a digital picture? Well, there are many, many places to find Ruby help. Unfortunately, that's sort of unhelpful, isn't it? 😃

For me, there are really only three places I look for Ruby help. If it's a small question, and I think I can experiment on my own to find the answer, I use irb. If it's a bigger question, I look it up in my pickaxe. And if I just can't figure it out on my own, then I ask for help on ruby-talk.

IRB: Interactive Ruby

If you installed Ruby, then you installed irb. To use it, just go to your command prompt and type irb. When you are in irb, you can type in any ruby expression you want, and it will tell you the value of it. Type in 1 + 2, and it will tell you 3. (Note that you don't have to use puts.) It's kind of like a giant Ruby calculator. When you are done, just type in exit.

There's a lot more to irb than this, but you can learn all about it in the pickaxe.

The Pickaxe: "Programming Ruby"

Absolutely the Ruby book to get is "Programming Ruby, The Pragmatic Programmer's Guide", by David Thomas and Andrew Hunt (the Pragmatic Programmers). While I highly recommend picking up the 4th edition of this excellent book, with all of the latest Ruby covered, you can also get a slightly older (but still mostly relevant) version for free online.

You can find just about everything about Ruby, from the basic to the advanced, in this book. It's easy to read; it's comprehensive; it's just about perfect. I wish every language had a book of this quality. At the back of the book, you'll find a huge section detailing every method in every class, explaining it and giving examples. I just love this book!

There are a number of places you can get it (including the Pragmatic Programmers' own site), but my favorite place is at ruby-doc.org. That version has a nice table of contents on the side, as well as an index. (ruby-doc.org has lots of other great documentation as well, such as for the Core API and Standard Library... basically, it documents everything Ruby comes with right out of the box. Check it out.)

And why is it called "the pickaxe"? Well, there's a picture of a pickaxe on the cover of the book. It's a silly name, I guess, but it stuck.

Ruby-Talk: the Ruby Mailing List

Even with irb and the pickaxe, sometimes you still can't figure it out. Or perhaps you want to know if someone already did whatever it is you are working on, to see if you could use it instead. In these cases, the place to go is ruby-talk, the Ruby Mailing List. It's full of friendly, smart, helpful people. To learn more about it, or to subscribe, look here.

WARNING: There's a lot of mail on the mailing list every day. I have mine automatically sent to a different mail folder so that it doesn't get in my way. If you just don't want to deal with all that mail, though, you don't have to! The ruby-talk mailing list is mirrored to the newsgroup comp.lang.ruby, and vice versa, so you can see the same messages there. Either way, you see the same messages, just in a slightly different format.

Tim Toady

Something I have tried to shield you from, but which you will surely run in to soon, is the concept of TMTOWTDI (pronounced "Tim Toady"): There's More Than One Way To Do It.

Now some will tell you what a wonderful thing TMTOWTDI is, while others feel quite differently. I don't really have strong feelings about it in general, but I think it's a terrible way to teach someone how to program. (As if learning one way to do something wasn't challenging and confusing enough!)

However, now that you are moving beyond this tutorial, you'll be seeing much more diverse code. For example, I can think of at least five other ways to make a string (aside from surrounding some text in single quotes), and each one works slightly differently. I only showed you the simplest of the six.

And when we talked about branching, I showed you if, but I didn't show you unless. I'll let you figure that one out in irb.

Another nice little shortcut you can use with if, unless, and while, is the cute one-line version:

```
# These words are from a program I wrote to generate
# English-like babble.  Cool, huh?
puts 'grobably combergearl thememberate' if 5 == 2**2 + 1**1
puts 'enlestrationshifter supposine' unless 'Chris'.length == 5
```

```
grobably combergearl thememberate
```

And finally, there is another way of writing methods which take blocks (not procs). We saw the thing where we grabbed the block and turned it into a proc using the &block trick in your parameter list when you define the function. Then, to call the block, you just use block.call. Well, there's a shorter way (though I personally find it more confusing). Instead of this:

```
def doItTwice(&block)
  block.call
  block.call
end

doItTwice do
  puts 'murditivent flavitemphan siresent litics'
end
```

```
murditivent flavitemphan siresent litics
murditivent flavitemphan siresent litics
```

...you do this:

```
def doItTwice
  yield
  yield
end

doItTwice do
  puts 'buritiate mustripe lablic acticise'
end
```

```
buritiate mustripe lablic acticise
buritiate mustripe lablic acticise
```

I don't know... what do you think? Maybe it's just me, but... yield?! If it was something like call_the_hidden_block or something, that would make a lot more sense to me. A lot of people say yield makes sense to them. But I guess that's what TMTOWTDI is all about: they do it their way, and I'll do it my way.

THE END

Use it for good and not evil. 😃 And if you found this tutorial useful (or confusing, or if you found an error), let me know!Beyond This Tutorial