11. Что не вошло в учебник

Итак, куда мы сейчас отправимся? Кого можно спросить, если у вас возник вопрос? А если вы хотите, чтобы ваша программа открывала web-страничку, отправляла электронную почту или масштабировала цифровое изображение? Что ж, есть полным-полно мест, где найдется помощь по Ruby. Но, к сожалению, такой ответ вам не слишком поможет, не так ли? :-)

Что касается меня, то на самом деле есть только три места, где я ищу помощи по Ruby. Если это небольшой вопрос, и я полагаю, что я могу сам поэкспериментировать, чтобы найти на него ответ, то я использую irb. Если это вопрос посерьёзнее, я обращаюсь к своей киркомотыге. А если я никак не могу разобраться с ним сам, то я прошу помощи в ruby-talk.

IRB: интерактивный интерпретатор Ruby

Если вы установили Ruby, то вы также установили `irb`. Чтобы его запустить, нужно просто перейти в командное окно и напечатать `irb`. Когда вы находитесь в сеансе `irb`, вы можете вводить любые выражения языка Ruby, какие пожелаете, а он будет выдавать вам их значения. Введите `1 + 2`, и он выдаст вам 3. (Обратите внимание, что вам не нужно использовать **`puts`**.) Это похоже на гигантский калькулятор на Ruby. Когда вы закончите, просто введите команду `exit`.

В `irb` имеется гораздо больше этого, но обо всём этом вы можете узнать в "киркомотыге".

Киркомотыга: "Программирование на Ruby"

Абсолютно *именно та* книга по Ruby, которая нужна всем — это "Программирование на Ruby: Руководство прагматичного программиста", написанная Дэвидом Томасом и Эндрю Хантом (Программисты-прагматики). Хотя я очень рекомендую достать 2-е издание этой замечательной книги, где освещены все последние возможности Ruby, вы также можете взять немного более старую (но до сих пор в основном подходящую) версию, бесплатно доступную в Сети. (На самом деле, если вы установили версию Ruby для Windows, то она у вас уже имеется.)

В этой книге вы можете найти о руби Ruby почти всё, от основ до самых передовых возможностей. Она легко читается; она исчерпывающая; она почти бузупречна. Хотелось бы, чтобы для всякого языка имелась бы книга подобного качества. В конце книги вы найдёте огромный раздел, где подробно описан каждый метод каждого класса, приведены объяснения и примеры. Я просто полюбил эту книгу!

Её можно найти в нескольких местах (включая собственный сайт Программистов-прагматиков), но мне больше всего нравится сайт ruby-doc.org. В этой версии слева имеется симпатичное оглавление, а также предметный указатель. (На ruby-doc.org есть также много другой документации, например, о базовом API и стандартной

библиотеке... В основном, там есть готовые к использованию документы обо всём, что касается Ruby. Проверьте сами.)

А почему же она называется "киркомотыга" ("the pickaxe")? Ну, там на обложке книги есть картинка киркомотыги. Мне кажется, это глупое название, но оно уже "прилипло".

Ruby-Talk: список рассылки о Ruby

Даже имея irb и *киркомотыгу*, вы иногда всё-таки не можете разобраться с чем-нибудь. Или, возможно, вы хотите знать, не делал ли уже кто-нибудь то, над чем вы сейчас работаете, чтобы выяснить, можно ли вам этим воспользоваться. В этих случаях вам нужно обратиться именно в ruby-talk, список рассылки о Ruby. В нём полно дружелюбных, умных, отзывчивых людей. Чтобы побольше узнать о нём или подписаться на него, взгляните сюда.

ПРЕДУПРЕЖДЕНИЕ: В этой рассылке каждый день приходит *много* почты. У меня она автоматически пересылается в другой почтовый ящик, поэтому она мне не мешает. Если же вы не желаете иметь дело со всей этой почтой, то это вам и не нужно! Список рассылки ruby-talk зеркалируется в новостную группу comp.lang.ruby, и наоборот, так что там вы можете увидеть те же самые сообщения. Любым из этих способов вы увидите одни и те же сообщения, просто немного в разных форматах. [Web-интерфейс к архиву этого списка рассылки находится по адресу: www.ruby-forum.com/forum/4. — *Прим. перев.*]

Tim Toady

То, от чего я старался уберечь вас, но с чем вы непременно скоро столкнётесь, это принцип TMTOWTDI (произносится "Тим Тоуди") или "There's More Than One Way To Do It", что значит "Есть не один способ сделать что-либо".

В то время как одни будут говорить вам, какая замечательная вещь этот TMTOWTDI, другие относятся к нему совсем по-другому. На самом деле, у меня в основном не возникает по этому поводу никаких сильных эмоций, но я считаю, что это *ужасный* метод обучить кого-то, как нужно программировать. (Как будто научить делать что-то одним способом это само по себе не достаточно трудоёмкое и сложное дело!)

Однако теперь, когда вы выходите за рамки того учебника, вы будете читать гораздо более разнообразные программы. Например, мне приходят на ум по крайней мере пять различных способов создания строки (помимо заключения некоторого текста в одинарные кавычки), и каждый из них работает немного по-другому. Я показал вам только самый простой из этих шести.

А когда мы говорили о ветвлении, я показал вам **`if`**, но не показал **`unless`**. Предоставляю вам выяснить что это такое в `irb`.

Ещё одно приятное сокращение, которое вы можете использовать для **`if`**, **`unless`** и **`while`**, это симпатичная однострочная версия:

```
#  Эти слова взяты из программы, которую я написал для генерирования
```

```
#  англоподобной болтовни.  Круто, да?
puts 'grobably combergearl kitatently thememberate' if 5 == 2**2 + 1**1
puts 'enlestrationshifter supposine follutify blace' unless 'Chris'.length ==
5
```

```
grobably combergearl kitatently thememberate
```

И наконец, есть ещё один способ писать методы, которые принимают блоки (а не процедурные объекты). Мы видели это, когда мы захватывали блок и превращали его в процедурный объект, используя трюк с **`&block`** в списке параметров при определении функции. Тогда, чтобы вызвать блок, вы просто используете **`block.call`**. Ладно, есть способ покороче (хотя лично я нахожу его более запутанным). Вместо этого:

```
def doItTwice(&block)
  block.call
  block.call
end
doItTwice do
  puts 'murditivent flavitemphan siresent litics'
end
```

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

...вы делаете так:

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

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

Ну, не знаю... а что вы об этом думаете? Возможно, так только я так считаю, но... **`yield`**?! Если бы это было что-нибудь наподобие **`call_the_hidden_block`** или что-то в этом роде, что бы имело *немного* больше смысла для меня. Многие люди говорят, что **`yield`** для них имеет смысл. Но я думаю, что принцип TMTOWTDI предполагает вот что: они делают что-то по-своему, а я делаю это по-моему.

КОНЕЦ

Используйте всё это во благо, а не во зло. :-) И если вы находите этот учебник полезным (или запутанным, либо если вы нашли ошибку), дайте мне знать!