Learning Ruby - 11

Grabbag – A Bunch of Other Stuff

Ruby Naming Conventions

- Diagram taken from page 17 of *The PickAxe.*

Table 2.1. Example variable and class names

Variables				Constants and
Local	**Global**	**Instance**	**Class**	**Class Names**
name	$debug	@name	@@total	PI
fish_and_chips	$CUSTOMER	@point_1	@@symtab	FeetPerMile
x_axis	$_	@X	@@N	String
thx1138	$plan9	@_	@@x_pos	MyClass
_26	$Global	@plan9	@@SINGLE	JazzSong

Operator Expressions

- Diagram taken from page 339 of *The Pickaxe*.

Table 22.4. Ruby operators (high to low precedence)

Method	Operator	Description
✓	`[ ] [ ]=`	Element reference, element set
✓	`**`	Exponentiation
✓	`! ~ + -`	Not, complement, unary plus and minus (method names for the last two are +@ and -@)
✓	`* / %`	Multiply, divide, and modulo
✓	`+ -`	Plus and minus
✓	`>> <<`	Right and left shift
✓	`&`	"And" (bitwise for integers)
✓	`^ |`	Exclusive "or" and regular "or" (bitwise for integers)
✓	`<= < > >=`	Comparison operators
✓	`<=> == === != =~ !~`	Equality and pattern match operators (!= and !~ may not be defined as methods)
	`&&`	Logical "and"
	`||`	Logical "or"
	`.. ...`	Range (inclusive and exclusive)
	`? :`	Ternary if-then-else
	`= %= ~= /= -= += |= &= >>= <<= *= &&= ||= **=`	Assignment
	`defined?`	Check if symbol defined
	`not`	Logical negation
	`or and`	Logical composition
	`if unless while until`	Expression modifiers
	`begin/end`	Block expression

Fun with `backticks`

```
`date`

`ls`
`ls`.split
`ls`.split[-1]

`uptime`

puts $?

`ls rubbbbish`
puts $?
`ls /etc/passwd`
puts $?
```

Swapping Values

```
temp = a
a = b
b = temp
```

```
a, b = b, a
```

if and unless

```
if song.artist == "Crowded House" then
   song.play
elsif song.artist == "Snow Patrol" then
   song.hold
else
   song.delete!
end # of if.

unless song.artist == "Led Zeppelin"
   song.turn_up!( 20 )
else
   song.turn_down!( 20 )
end # of unless.
```

```
song.artist == "Led Zeppelin" ? song.turn_down!( 20 ) : song.turn_up!( 20 )
```

`if` Assignments

```
lead_singer =  if song.artist == "Crowded House"
                  "Neil Finn"
               elsif song.artist == "Led Zeppelin"
                  "Robert Plant"
               elsif song.artist == "The Beatles"
                  "John, Paul, John or Ringo"
               else
                  "unknown"
               end # of if.
```

Statement Qualifiers

```
# Borrowed from page 98 of The PickAxe.

File.foreach( "/etc/fstab" ) do |line|

   # Skip comments.
   next if line =~ /^#/

   # Parse line if it's not empty.
   parse( line ) unless line =~ /^$/

end # of do.
```

The Case for Case

```
album_title = case year
   when 1969 then "Led Zeppelin and Led Zeppelin II"
   when 1970 then "Led Zeppelin III"
   when 1971 then "(unnamed)"
   when 1973 then "Houses of the Holy"
   when 1975 then "Physical Graffiti"
   when 1976 then "Presence and The Song Remains The Same"
   when 1979 then "In Through The Out Door"
   else "unknown"
end # of case.
```

What's All This `then`?

- With **`if`** and **`case`**, the **`then`** keyword is (sometimes) optional
- When the expression is on the same line as the condition, the **`then`** is required
- When it isn't, well ... it isn't
- The word **`then`** can be replaced with **`:`** when used on the same line:

```
album_title = case year
          when 1969 : "Led Zeppelin and Led Zeppelin II"
          when 1970 : "Led Zeppelin III"
                           ...
```

Loops

- Ruby supports **`while`**, **`until`** and **`for`**
- Ruby also supports iterators (as we've already seen) which are just like loops
- Loops also support **`break`**, **`next`**, **`redo`** and **`retry`**
- There's also a **`loop`** loop:

```
loop do

     # This code will execute forever ...

end # of loop.
```

There's Much More to Ruby

- Chapter 22 of *The PickAxe* is the concise reference to the Ruby Language ... take the time to browse these 45 pages
- In fact, pick up a copy of *The PickAxe* and keep it by your side while programming Ruby
- And remember ... Ruby is known as a **fun programming language** ... *it's quite OK to have fun while programming (imagine that!)*