Ruby Essentials
# Table of Contents

Chapter 1. About Ruby Essentials

Chapter 2. What is Ruby?

  2.1 The History of Ruby
  2.2 What is Ruby?
  2.3 Why is Ruby so Popular?

Chapter 3. Getting and Installing Ruby

  3.1 Installing Ruby on Linux
    3.1.1 Ruby on Red Hat Enterprise and Fedora Linux
    3.1.2 Ruby on Ubuntu and Debian Linux
    3.1.3 Ruby on Microsoft Windows

Chapter 4. Simple Ruby Examples

  4.1 The Most Basic Ruby Example
  4.2 Executing Ruby from the Command Line
  4.3 Interactive Ruby Execution
  4.4 Executing Ruby from a File
  4.5 Creating a Self Contained Ruby Executable on Linux or UNIX
  4.6 Associating Ruby Files on Windows

Chapter 5. Commenting Ruby Code

  5.1 What exactly is Commenting
  5.2 Single Line Ruby Comments
  5.3 Comments on Lines of Code
  5.4 Multi Line or Block Ruby Comments

Chapter 6. Understanding Ruby Variables

  6.1 Ruby Constants
  6.2 Ruby and Variable Dynamic Typing
  6.3 Declaring a Variable
  6.4 Identifying a Ruby Variable Type
Chapter 6.

6.5 Changing Variable Type

6.6 Converting Variable Values

Chapter 7. Ruby Variable Scope

7.1 What is Variable Scope?

7.2 Detecting the Scope of a Ruby Variable

7.3 Ruby Local Variables

7.4 Ruby Global Variables

7.5 Ruby Class Variables

7.6 Ruby Instance Variables

7.7 Ruby Constant Scope

Chapter 8. Ruby Number Classes and Conversions

8.1 Ruby Number Classes

8.1.1 Integer Class

8.1.2 Fixnum Class

8.1.3 Bignum Class

8.1.4 Float Class

8.1.5 Rational Class

8.2 Converting Numbers in Ruby

8.2.1 Convert Floating Point Number to an Integer

8.2.2 Convert a String to an Integer

8.2.3 Convert a Hexadecimal Number to an Integer

8.2.4 Convert an Octal Number to an Integer

8.2.5 Convert a Binary Number to an Integer

8.2.6 Convert a Character to the ASCII Character Code

8.2.7 Convert an Integer Floating Point

8.2.8 Convert a String to Floating Point

8.2.9 Convert a Hexadecimal Number to Floating Point

8.2.10 Convert an Octal Number to a Floating Point

© 2009 Techotopia.com
12.7 Deleting Array Elements

12.8 Sorting Arrays

Chapter 13. Ruby Operators

13.1 The Anatomy of a Ruby Operation

13.2 Performing Ruby Arithmetic using Operators

13.3 Ruby Assignment Operators

13.4 Parallel Assignment

13.5 Ruby Comparison Operators

13.6 Ruby Bitwise Operators

13.7 Summary

Chapter 14. Ruby Operator Precedence

14.1 An Example of Ruby Operator Precedence

14.2 Overriding Operator Precedence

Chapter 15. Ruby Math Functions and Methods

15.1 Ruby Math Constants

15.2 Ruby Math Methods

15.3 Some Examples

15.4 Summary

Chapter 16. Understanding Ruby Logical Operators

16.1 Ruby Logical Operators

Chapter 17. Ruby Object Oriented Programming

17.1 What is an Object?

17.2 What is a Class?

17.3 Defining a Ruby Class

17.4 Creating an Object from a Class

17.5 Instance Variables and Accessor Methods

17.6 Ruby Class Variables
23.1 Concatenating Strings in Ruby ................................................................. 94
23.2 Freezing a Ruby String .............................................................................. 94
23.3 Accessing String Elements ...................................................................... 95
23.4 Comparing Ruby Strings .......................................................................... 96
23.5 Case Insensitive String Comparisons ....................................................... 97

Chapter 24. Ruby String Replacement, Substitution and Insertion ................. 98
24.1 Changing a Section of a String .................................................................. 98
24.2 Ruby String Substitution .......................................................................... 99
24.3 Repeating Ruby Strings ........................................................................... 99
24.4 Inserting Text into a Ruby String ............................................................. 100
24.5 Ruby chomp and chop Methods ............................................................... 100
24.6 Reversing the Characters in a String ....................................................... 101

Chapter 25. Ruby String Conversions ............................................................ 102
25.1 Converting a Ruby String to an Array .................................................... 102
25.2 Changing the Case of a Ruby String ....................................................... 103
25.3 Performing String Conversions ............................................................... 103
25.4 Summary .................................................................................................. 104

Chapter 26. Ruby Directory Handling ............................................................ 105
26.1 Changing Directory in Ruby ...................................................................... 105
26.2 Creating New Directories ......................................................................... 105
26.3 Directory Listings in Ruby ........................................................................ 105
26.4 Summary .................................................................................................. 106

Chapter 27. Working with Files in Ruby .......................................................... 107
27.1 Creating a New File with Ruby ................................................................. 107
27.2 Opening Existing Files ............................................................................. 107
27.3 Renaming and Deleting Files in Ruby ...................................................... 108
27.4 Getting Information about Files ............................................................... 108
27.5 Reading and Writing Files ....................................................................... 110
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>28.1</td>
<td>Accessing the Date and DateTime Classes in Ruby</td>
<td>113</td>
</tr>
<tr>
<td>28.2</td>
<td>Working with Dates in Ruby</td>
<td>113</td>
</tr>
<tr>
<td>28.3</td>
<td>Working with Dates and Times</td>
<td>114</td>
</tr>
<tr>
<td>28.4</td>
<td>Calculating the Difference Between Dates</td>
<td>114</td>
</tr>
</tbody>
</table>
Chapter 1. About Ruby Essentials

Ruby is a flexible and intuitive object-oriented programming language. From modest beginnings in Japan where it rapidly gained a loyal following, the popularity of Ruby has now spread throughout the programming world.

This surge in popularity can, in no small part, be attributed to the introduction and wide adoption of the Ruby on Rails framework. It is difficult, however, to get the most out of Ruby on Rails without first learning something about programming in Ruby, and this is where Ruby Essentials comes in.

Ruby Essentials is intended to provide a concise and easy to follow guide to learning Ruby. Everything from installing Ruby and the basics of the language through to topics such as arrays, file handling and object-oriented programming are covered, all combined with easy to understand code examples which serve to bridge the gap between theory and practice.

Ruby Essentials is designed to be of equal use both to those experienced in other programming languages and to novices who have chosen Ruby as their "first programming language".
Chapter 2. What is Ruby?

In this chapter of Ruby Essentials we will learn about what Ruby is, how it came into existence and what it is useful for.

2.1 The History of Ruby

Ruby was created by Yukihiro Matsumoto (more affectionately known as Matz) in Japan starting in 1993. Matz essentially kept Ruby to himself until 1995 when he released it to the public. Ruby quickly gained a following in Matz's home country of Japan in the following years, and finally gained recognition in the rest of the programming world beginning in the year 2000. From that point on Ruby has grown in popularity, particularly because of the popularity of the Ruby on Rails web application development framework.

2.2 What is Ruby?

Ruby is an object-oriented interpreted scripting language. When we say it is interpreted we mean to say that the Ruby source code is compiled by an interpreter at the point of execution (similar in this regard to JavaScript and PHP). This contrasts with compiled languages such as C or C++ where the code is pre-compiled into a binary format targeted to run on a specific brand of microprocessor.

There are advantages and disadvantages to being an interpreted language. A disadvantage is speed. Because the source code has to be interpreted at runtime this means that it runs slower than an equivalent compiled application. A secondary concern for some is the fact that anyone who uses your application will also be able to see the source code. In the world of open source this is less of a problem than it used to be, but for some proprietary applications this might prove to be unacceptable.

The primary advantage of interpreted languages is that they are portable across multiple operating system platforms and hardware architectures. A compiled application, on the other hand, will only run on the operating system and hardware for which it was compiled. For example, you can take a Ruby application and run it without modification on an Intel system running Linux, an Intel system running Windows, an Intel system running Mac OS X or even a PowerPC system running Mac OS or Linux. To do this with a C or C++ application you would need to compile the code on each of the 5 different systems and make each binary image available. With Ruby you just supply the source code.

Another advantage of being interpreted is that we can write and execute Ruby code in real-time directly in the Ruby interpreter. For those who like to try things out in real time (and not everyone does), this is an invaluable feature.
2.3 Why is Ruby so Popular?

Firstly, Ruby is a very intuitive and clean programming language. This makes learning Ruby a less challenging task than learning some other languages. Ruby is also a great general purpose language. It can be used to write scripts in the same way you might use Perl and it can be used to create full scale, standalone GUI based applications. Ruby's usefulness doesn't end there however. Ruby is also great for serving web pages, generating dynamic web page content and excels at database access tasks.

Not only is Ruby intuitive and flexible it is also extensible, enabling new functionality to be added through the integration third-party, or even home grown libraries.

And, of course, being an interpreted language means that Ruby is portable. Once an application has been developed in Ruby it will run equally well on Ruby supported platforms such as Linux, UNIX, Windows and Mac OS X.