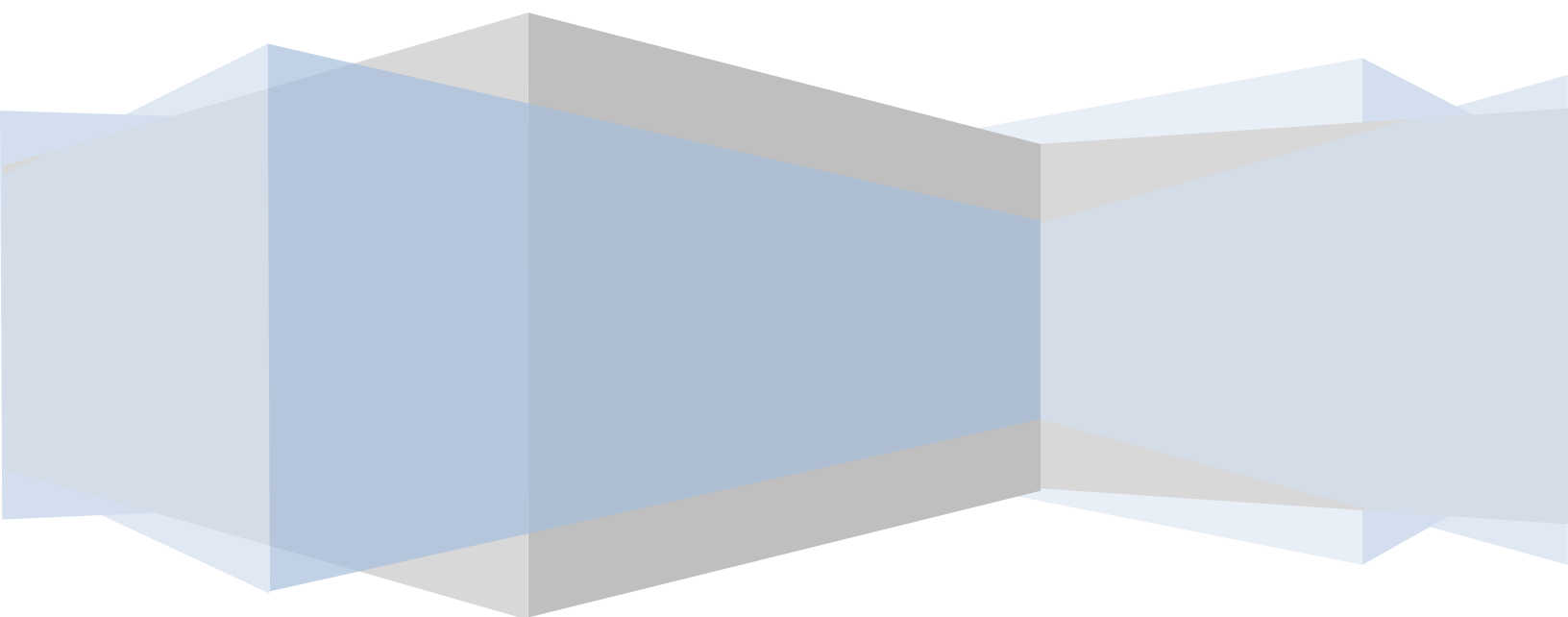


Ruby Essentials



Ruby Essentials – First Edition

© 2009 Techotopia.com. This eBook is provided for personal use only. Unauthorized use, reproduction and/or distribution strictly prohibited. All rights reserved.

Table of Contents

Chapter 1.	About Ruby Essentials.....	10
Chapter 2.	What is Ruby?	11
2.1	The History of Ruby	11
2.2	What is Ruby?.....	11
2.3	Why is Ruby so Popular?	12
Chapter 3.	Getting and Installing Ruby.....	13
3.1	Installing Ruby on Linux.....	13
3.1.1	Ruby on Red Hat Enterprise and Fedora Linux	13
3.1.2	Ruby on Ubuntu and Debian Linux	14
3.1.3	Ruby on Microsoft Windows	15
Chapter 4.	Simple Ruby Examples	18
4.1	The Most Basic Ruby Example	18
4.2	Executing Ruby from the Command Line.....	19
4.3	Interactive Ruby Execution	19
4.4	Executing Ruby from a File.....	21
4.5	Creating a Self Contained Ruby Executable on Linux or UNIX	21
4.6	Associating Ruby Files on Windows	22
Chapter 5.	Commenting Ruby Code	25
5.1	What exactly is Commenting	25
5.2	Single Line Ruby Comments.....	25
5.3	Comments on Lines of Code	26
5.4	Multi Line or Block Ruby Comments.....	26
Chapter 6.	Understanding Ruby Variables	27
6.1	Ruby Constants.....	27
6.2	Ruby and Variable Dynamic Typing.....	27
6.3	Declaring a Variable	28
6.4	Identifying a Ruby Variable Type	28

6.5	Changing Variable Type.....	29
6.6	Converting Variable Values	29
Chapter 7.	Ruby Variable Scope	31
7.1	What is Variable Scope?.....	31
7.2	Detecting the Scope of a Ruby Variable.....	31
7.3	Ruby Local Variables.....	32
7.4	Ruby Global Variables	32
7.5	Ruby Class Variables.....	33
7.6	Ruby Instance Variables	33
7.7	Ruby Constant Scope.....	34
Chapter 8.	Ruby Number Classes and Conversions.....	35
8.1	Ruby Number Classes.....	35
8.1.1	Integer Class.....	35
8.1.2	Fixnum Class.....	35
8.1.3	Bignum Class	35
8.1.4	Float Class	35
8.1.5	Rational Class	35
8.2	Converting Numbers in Ruby	36
8.2.1	Convert Floating Point Number to an Integer	36
8.2.2	Convert a String to an Integer.....	36
8.2.3	Convert a Hexadecimal Number to an Integer	36
8.2.4	Convert an Octal Number to an Integer	36
8.2.5	Convert a Binary Number to an Integer	36
8.2.6	Convert a Character to the ASCII Character Code	36
8.2.7	Convert an Integer Floating Point.....	37
8.2.8	Convert a String to Floating Point.....	37
8.2.9	Convert a Hexadecimal Number to Floating Point	37
8.2.10	Convert an Octal Number to a Floating Point.....	37

8.2.11	Convert a Binary Number to Floating Point.....	37
8.2.12	Convert an Character to a Floating Point ASCII Character Code	37
Chapter 9.	Ruby Methods.....	38
9.1	Declaring and Calling a Ruby Method	38
9.2	Passing Arguments to a Method	38
9.3	Passing a Variable Number of Arguments to a Method	39
9.4	Returning a Value from a Function	40
9.5	Ruby Method Aliases.....	40
Chapter 10.	Ruby Ranges.....	42
10.1	Ruby Sequence Ranges	42
10.2	Using Range Methods	43
10.3	Ruby Ranges as Conditional Expressions	44
10.4	Ruby Range Intervals.....	44
10.5	Ranges in case Statements.....	44
Chapter 11.	Understanding Ruby Arrays	46
11.1	What is a Ruby Array.....	46
11.2	How to Create a Ruby Array.....	46
11.3	Populating an Array with Data	47
11.4	Finding Out Information about a Ruby Array.....	47
11.5	Accessing Array Elements	48
11.6	Finding the Index of an Element	49
Chapter 12.	Advanced Ruby Arrays	50
12.1	Combining Ruby Arrays	50
12.2	Intersection, Union and Difference.....	50
12.3	Identifying Unique Array Elements	52
12.4	Pushing and Popping Array Elements	52
12.5	Ruby Array Comparisons.....	53
12.6	Modifying Arrays	53

12.7	Deleting Array Elements.....	54
12.8	Sorting Arrays.....	55
Chapter 13.	Ruby Operators.....	57
13.1	The Anatomy of a Ruby Operation.....	57
13.2	Performing Ruby Arithmetic using Operators.....	57
13.3	Ruby Assignment Operators	58
13.4	Parallel Assignment.....	59
13.5	Ruby Comparison Operators.....	59
13.6	Ruby Bitwise Operators.....	60
13.7	Summary	61
Chapter 14.	Ruby Operator Precedence.....	62
14.1	An Example of Ruby Operator Precedence.....	62
14.2	Overriding Operator Precedence	62
14.3	Operator Precedence Table	62
Chapter 15.	Ruby Math Functions and Methods	64
15.1	Ruby Math Constants.....	64
15.2	Ruby Math Methods	64
15.3	Some Examples	65
15.4	Summary	65
Chapter 16.	Understanding Ruby Logical Operators	66
16.1	Ruby Logical Operators	66
Chapter 17.	Ruby Object Oriented Programming	68
17.1	What is an Object?	68
17.2	What is a Class?	68
17.3	Defining a Ruby Class	68
17.4	Creating an Object from a Class.....	69
17.5	Instance Variables and Accessor Methods	69
17.6	Ruby Class Variables.....	71

17.7	Instance Methods.....	72
17.8	Ruby Class Inheritance	73
Chapter 18.	Ruby Flow Control.....	75
18.1	The Ruby if Statement.....	75
18.2	Using else and elsif Constructs.....	76
18.3	The Ruby Ternary Operator	77
18.4	Summary	78
Chapter 19.	The Ruby case Statement	79
19.1	Ruby Flow Control	79
19.2	Number Ranges and the case statement.....	81
19.3	Summary	81
Chapter 20.	Ruby While and Until Loops.....	82
20.1	The Ruby While Loop	82
20.2	Breaking from While Loops	83
20.3	unless and until	83
20.4	Summary	84
Chapter 21.	Looping with for and the Ruby Looping Methods	85
21.1	The Ruby for Loop	85
21.2	The Ruby times Method.....	87
21.3	The Ruby upto Method	87
21.4	The Ruby downto Method	88
Chapter 22.	Ruby Strings - Creation and Basics.....	89
22.1	Creating Strings in Ruby	89
22.2	Quoting Ruby Strings.....	89
22.3	General Delimited Strings	90
22.4	Ruby Here Documents	91
22.5	Getting Information about String Objects	92
Chapter 23.	Ruby String Concatenation and Comparison.....	94

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

Chapter 28. Working with Dates and Times in Ruby 113

 28.1 Accessing the Date and DateTime Classes in Ruby 113

 28.2 Working with Dates in Ruby 113

 28.3 Working with Dates and Times 114

 28.4 Calculating the Difference Between Dates 114

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.