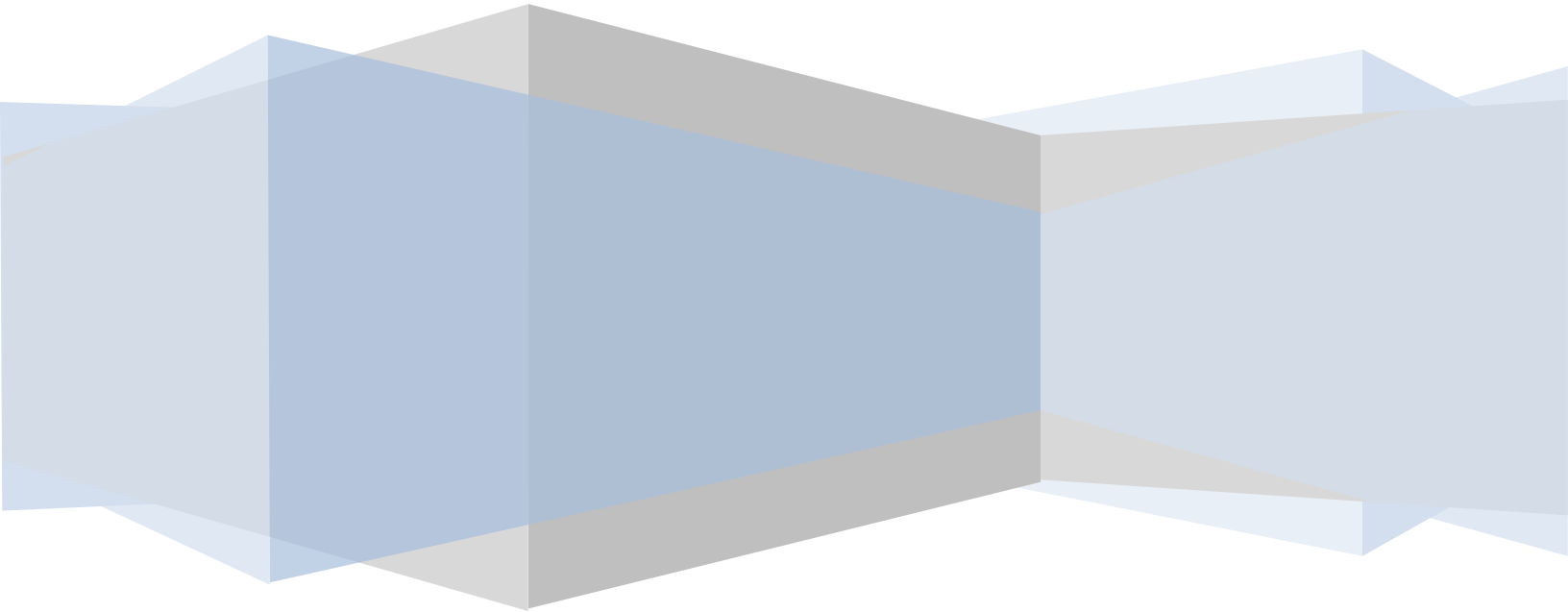
Ruby Essentials



Ruby Essentials – First Edition

Table of Contents

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.