Rebol is designed for the exchange and interpretation of information.

Rebol provides an easy-to-write, easy-to-read format and a wide variety of data-types for expressing data and code.

If you're familiar with JSON, then you know a simple easy way to share information between systems and processes. Rebol is like an extended version of JSON that provides numerous other lexical data-types and can also execute data as code.

Announcement (March 2024)

This Rebol website is moving to a new server that will provide HTTPS secure access via Apache. This is mainly to improve search engine indexing which penalizes non HTTPS websites. Please take note of the following:

- The www.rebol.com server will operate HTTP on port 80 without redirection. This will allow scripts that use that URL to continue to operate.
- CGI scripts for www.rebol.com are currently disabled. We will be re-enabling the important ones as time permits. However, CGI on rebol.com is still operating.
- Also, plans are in the works to improve CSS for pages that don't render well on mobile devices.

Interesting Facts...

- **Rebol is pronounced "reb-ol".**
  It's similar to "rebel yell" or "rebel with a cause".
- **Rebol rebels against the idea that modern software must be large and complex.**
  Such systems are painfully slow and expensive to develop and maintain. They quickly degrade into chaos, requiring constant updates and patches.
- **Rebol has been downloaded over 5 million times.**
  Many of its features have been added to other computer languages, but due to their older, syntax-driven designs, they will never come close to the expressive power of Rebol.
- **Rebol is nearly syntax free.**
  And, as you become fluent, you end up writing code in *sentences*, somewhat similar to human languages. Because our brains are well optimized for sequences of words, Rebol sentences feel natural, and you become more productive.
- **Rebol invented the concept of *dialects*.**
  Dialects provide expressive leverage not found in other languages. They amplify your coding effort by focusing on the essence of what you are trying to do, whether that's creating a user interface or programming a robot's motion.
- **Rebol is more than just a programming language.**
  It is also a language for representing data and metadata. It provides a consistent architecture for computation, storage, and exchange of information.
- **Rebol has been intentionally kept small and lightweight.**
  It only takes a few seconds to download, requires no special libraries, and includes hundreds of functions, dozens of data-types, built-in help, multiple Internet protocols, graphics, sound, and much more.
- **Rebol was designed by Carl Sassenrath**
  The operating system architect known for bringing multitasking to personal computers.

Interesting Examples...

- **Check out Just One Line**
  It gives you some idea of the power of Rebol by showing what's possible with just one line of code.
- **Or, check out the Demo Gallery or Reblet Gallery**
  They show what a few KB of Rebol data and code can do.
- **For more, see the Cookbook of Rebol Code Examples**
  It shows more detailed examples of using Rebol to solve specific problems.
- **Beyond that, visit the REBOL.org Library**
  Although a bit old, it shows hundreds of examples and useful programs.
- **See Tutorials and Examples**
  For many other links to code examples.

A Quick Example...

To get an idea what it looks like, take a simple digital clock as an example.

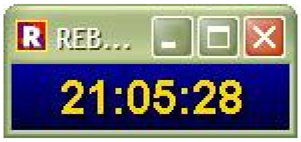
Every Rebol program begins with a header description. It's actually an object with fields that can be referenced from within the program or from other programs:

```
REBOL [
    Title:   "Digital Clock"
    Version: 1.3.3
    Author:  "Carl Sassenrath"
    Purpose: "Display a simple digital clock on screen."
]
```

The clock code itself is written in the Rebol VID dialect (Visual Interface Dialect). The code here describes a graphical layout that uses a text line (banner) that refreshes once a second, has a graphical gradient shading, and shows the new time every second:

```
clock: layout [
    origin 0
    clock-face: banner 140x32 rate 1
        effect [gradient 0x1 0.0.150 0.0.50]
        feel [engage: func [f a e] [set-face clock-face now/time]]
]
view clock
```

On screen, the clock looks like this:



21:05:28

If you want to see a list of core functions, check out docs/reference.html. Click on any link to see the details. For example check out the decompress function.