Red
programming language

Maxim Velesyuk

Features

red-lang.org

- **Homoiconic** (Red is its own meta-language and own **data-format**)
- **Functional**, imperative, **reactive** and **symbolic** programming
- **Prototype**-based object support
- **Gradual and multi-**typing
- **Macros** system
- **Rich** set of built-in datatypes (50+)
- **Cross-compilation** done right.
- Produces executables of **less than 1MB**, with **no dependencies.**
- **Concurrency** and **parallelism** strong support (actors, parallel collections)
- Low-level **system programming** abilities through the built-in Red/System DSL
- Powerful **PEG parser** DSL
- **Cross-platform native GUI** system, with a UI DSL and drawing DSL.
- **Low** memory footprint, **garbage collected**
- **Single-file** (~1MB) contains whole toolchain, standard library and REPL.
- **No install, no setup**
- **Fun** guaranteed. ;-)

Basics

Lisp: lists

```
CL-USER> (setf l '(1 2 3))
(1 2 3)
CL-USER> (type-of l)
CONS
```

Lisp: symbols

```
CL-USER> (type-of ‘l)
SYMBOL
```

Red: blocks

```
red>> b: [1 2 3]
== [1 2 3]
red>> type? b
== block!
```

Red: words

```
red>> type? 'b
== word!
```

Evaluation

Lisp: list is function call

```
CL-USER> (+ 1 2)
3
CL-USER> (eval ‘(+ 1 2))
3
```

Lisp: function definition

```
CL-USER> (defun f (a b) (+ a b))
F
```

Red: blocks are not evaluated

```
red>> [1 + 2]
== [1 + 2]
red>> do [1 + 2]
== 3
```

Red: function assignment

```
red>> f: func [a b] [a + b]
== func [a b][a + b]
```

Func is a function

red-lang.org

```
red>> help func

USAGE:
      func spec body

DESCRIPTION:
      Defines a function with a given spec and
body.
      func is of type: native!

ARGUMENTS:
     spec   [block!]
     body   [block!]
```

```
red>> args: [a b]
== [a b]
red>> body: [a + b]
== [a + b]
red>> f: func args body
== func [a b][a + b]
red>> f 1 2
== 3

red>> change find body-of :f '+ '-
== [b]
red>> f 1 2
== -1
red>> :f
== func [a b][a - b]
```

DSL / Dialect

red-lang.org

```
red>> 2 + 2 * 2
== 8
red>> (2 + 2) * 2
== 8

red>> math [2 + 2 * 2]
== 6
red>> source math


red>> length? split mold body-of :math newline
== 14
```

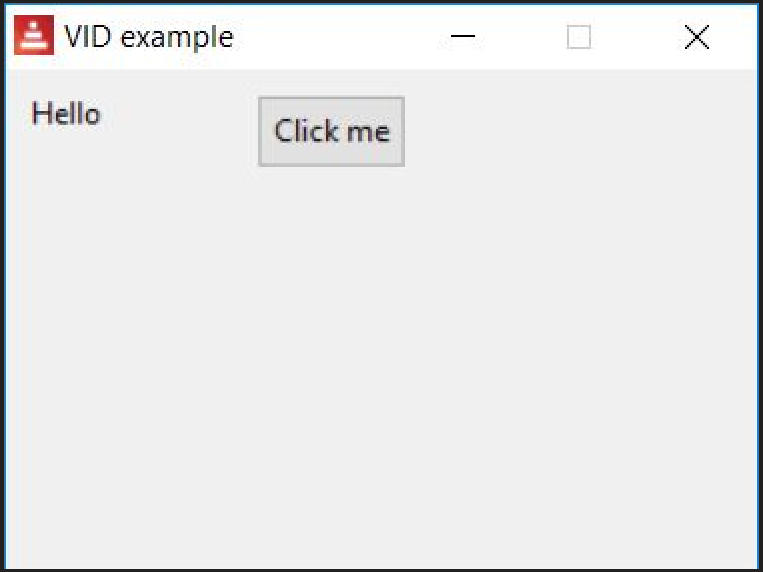
```
forth-math: func [code /local stack tmp w n] [
   stack: copy []
   parse code [
      any [
         set n number! (append stack n) |
         set w word! (
            tmp: take/part/last stack 2
            append stack do reduce [ tmp/1 w
tmp/2 ]
          )
      ]
   ]
   last stack
]

red>> forth-math [1 2 + 3 *]
== 9
red>> forth-math [2.9 1.4 -]
== 1.5
```

VID dialect

red-lang.org

```
red>> view [
      title "VID example"
      size 300x200
      text "Hello"
      button "Click me"
      ]
```



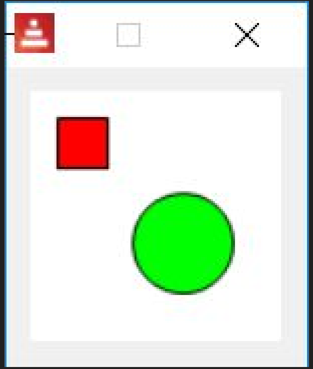
Draw dialect

red-lang.org

```
red>> i: make image! 100x100

red>> draw i [
      fill-pen red
      box 10x10 30x30
      fill-pen green
      circle 60x60 20
      ]

red>> view [image i]
```



What’s the catch?

Red’s current version is 0.6.2, released on 26, March.

It still lacks:

- Full I/O (planned in 0.7)
- Garbage collection (planned in 0.8)
- Concurrency
- Linux GUI support (you can play with graphics on Wine)
- Android support
- iOS support

Examples

```
>> ; get web page title
>> parse read https://booking.com [skip thru "<title>" copy title to "</title>"]
>> title
== {Booking.com: 1,227,392 hotels worldwide. 117+ million hotel reviews.}
```

```
>> ; remove all occurrences of element(s) from the list
>> list: [ x: z + 5 print x y: something print y ]
== [x: z + 5 print x y: something print y]
>> until [ not remove/part find list 'print 2 ]
>> list
== [x: z + 5 y: something]
```

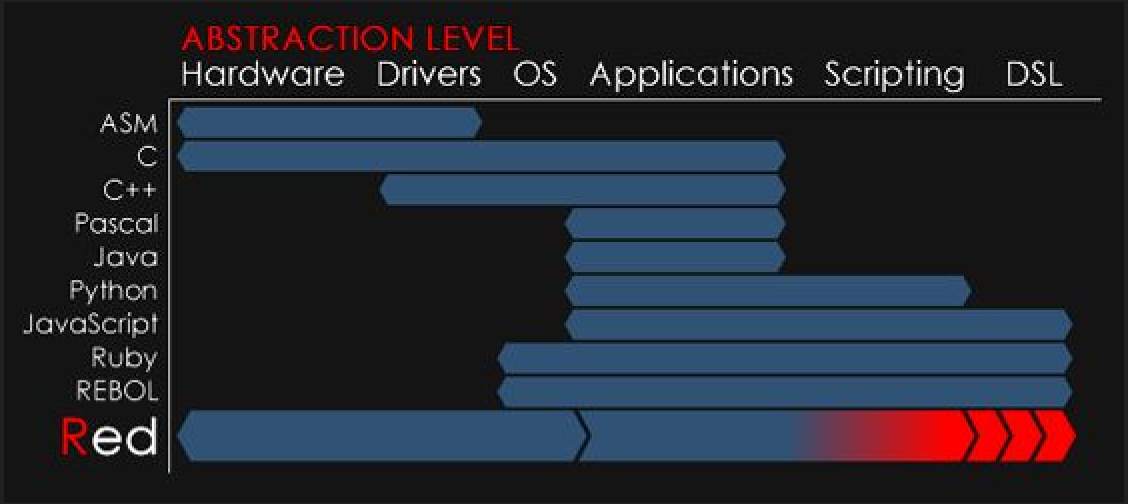
```
>> ; convert all bmp images in current directory to png
>> foreach file read %. [if find file %.bmp [save replace file %.bmp %.png load file]]
```

Live demo

red-lang.org

Full-stack programming language (c)

red-lang.org


ABSTRACTION LEVEL
Hardware
Drivers
OS
Applications
Scripting
DSL
ASM
C
C++
Pascal
Java
Python
JavaScript
Ruby
REBOL
Red