Rebol

Smarter
Faster
Better

Nordakademie, Elmshorn

18-Mai-2009

Robert M. Münch

www.robertmuench.de
robert.muench@robertmuench.de


Smarter, denn Rebol beinhaltet alles was man braucht:
„Mann muss eine Zeile Code schreiben können und ein Ergebnis sehen.“
SMARTER

Rebol läuft “out-of-the-box” wie der gute alte C-64.

Starten,

Coden,

Fertig.

```
rebol []

g: [
        style b box black 50x50
        style w b white space 0x0
]

loop 8 [
        append g head reverse/part [
                b w b w b w b w return
        ] 8
]

view layout g
```

GUI Inklusive

Schachbrett

Rebol läuft “out-of-the-box” wie der gute alte C-64.

Starten,

Coden,

Fertig.

```
rebol []

view f: layout for n 0 19 1 [
        r: (random 19) + n // 20

        append [across] load rejoin [
                "a"
                n
                ": check on [ a"
                r
                "/data: a"
                r
                "/data xor on show f ]"
        ]
]
```

GUI Inklusive

Spiel

Rebol läuft “out-of-the-box” wie der gute alte C-64.

Starten,

Coden,

Fertig.

```
remove-each tag page: load/markup
            http://www.rebol.com
            [tag? tag]

write %page.txt page
```

Netzwerk Inklusive

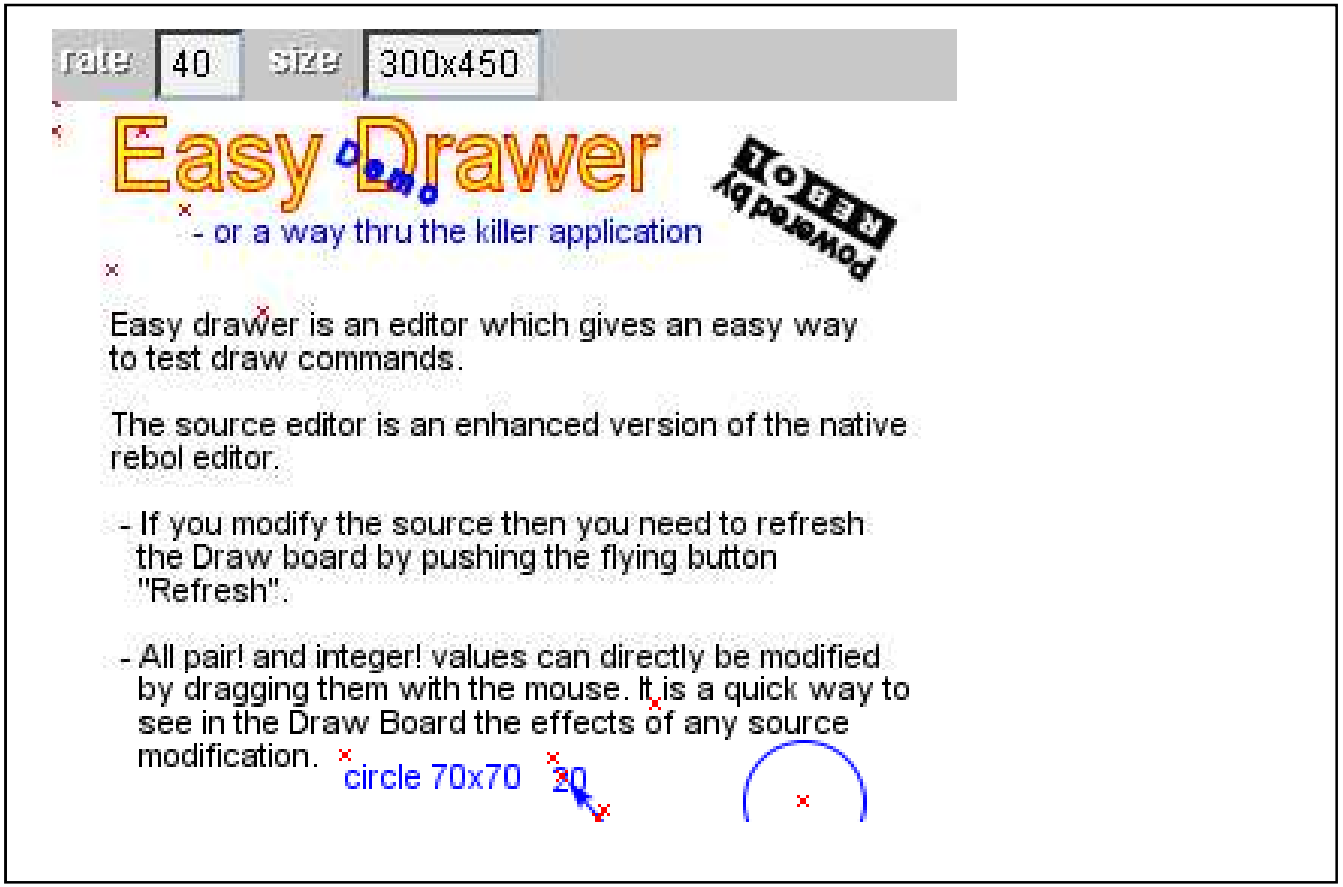
Web Seite als Text speichern

Rebol läuft “out-of-the-box” wie der gute alte C-64.

Starten,

Coden,

Fertig.



In 10 KB Code

kann man viel Funktionalität packen.

Rebol kennt keine Unterscheidung zwischen Daten und Code.

Alles erhält erst bei der Ausführung einen Context.

Es gibt nur ein Keyword: REBOL

Das **WORD** ist der Grundpfeiler

WORDshaben einen Typ:

- type? first [rebol] ; == word!
- type? first [rebol:] ; == set-word!
- type? first [:rebol] ; == get-word!
- type? first ['rebol] ; == lit-word!
- type? first [/rebol] ; == refinement!

Rebol kennt keine Unterscheidung zwischen Daten und Code.

Alles erhält erst bei der Ausführung einen Context.

Es gibt nur ein Keyword: REBOL

WORDs sind die Basis für Variablen

Zuweisungen:

- set 'rebolution "uprising"
- set/any 'rebolution "uprising"
- setfirst [rebolution:] "uprising"

Abfragen:

- :rebolution
- get 'rebolution
- get/any 'rebolution

Rebol kennt keine Unterscheidung zwischen Daten und Code.

Alles erhält erst bei der Ausführung einen Context.

Es gibt nur ein Keyword: REBOL

Ein **WORD** ist nur dann eine Variable, wenn es an einen **CONTEXT** gebunden ist.

Jedes **WORD** ist immer nur an einen **CONTEXT** gebunden.

Es gibt keine **CONTEXT** Hierarchie!

Es ist möglich zwei **WORD**s mit gleicher Schreibweise und Typ aber unterschiedlichem **BIND**ing zu definieren.

Rebol kennt keine Unterscheidung zwischen Daten und Code.

Alles erhält erst bei der Ausführung einen Context.

Es gibt nur ein Keyword: REBOL

```
code-blk: copy [a]
a: 12

make object! [append code-blk 'a a: 13]

>> equal? first code-blk second code-blk
== true

>> equal? bind? first code-blk bind? second code-blk
== false

>> code-blk
== [a a]
>> a
== 12
>> reduce code-blk
== [12 13]
```

write/binary %robertmuench.htmlread/binaryhttp://www.robertmuench.de

Rebol ist **schön** und **klar**.

Klarheit ist die Grundlage für gute Ergebnisse.

SAPHIRION

Integriertes Memory-Management

großer Wortschatz

Rebol ist der Duden
und nicht die BILD Zeitung
der Programmiersprachen.

Rebol ist **schön** und **klar**.

Klarheit ist die Grundlage für gute Ergebnisse.

SAPHIRION

Eine klare Syntax und Begrifflichkeit
ist die Voraussetzung für

gute Wartbarkeit und
Entwickelungen in großen Teams.

Rebol ist **schön** und **klar**.

Klarheit ist die Grundlage für gute Ergebnisse.

SAPHIRION


Faster, denn Rebol ist dicht und die perfekte GLUE Sprache.
FASTER

Rebol baut auf verschiedensten Dialekten auf.

(Domain SpecificLanguages)

Dialekt	Interpretiert von	Zweck
Data Exchange		Definiert TYPE und VALUE von LITERALs als Basis für alle weiteren Dialekte
Do	DO function	Programmierung
Parse	PARSE function	„Pattern Matching“ Grammatiken parsen, Dialekte
Function specification	make FUNC make FUNCTION	Funktionsdefinition Funtionale Programmierung
Object specification	make OBJECT	Objektdefinition Prototypen basierte Programmierung
Visual Interface	LAYOUTfunction	Grafisches User Interface
Draw	VIEW function	Grundlegende Zeichenformen wie Linien, Kreise, Polygone, Splines
Script specification	DO function	Script Defintion

Der „Data Exchange Dialect“ definiert für jedes „Literal“ den TYPE und VALUE und ist damit das bessere XML.

[http://www.google.com http://www.yahoo.com] fetch no-one@nowhere.com

block! word! email!

Rebol baut auf verschiedensten Dialekten auf.

(Domain Specific Languages)

SAPHIRION

Die Semantik wird rein durch den interpretierenden Dialekt bestimmt.
Literals des „Data Exchange Dialects“ werden somit zu Token.

Interpretation durch den DO Dialekt:

```
>> PRINT no-one@nowhere.com
== no-one@nowhere.com
```

Rebol baut auf verschiedensten Dialekten auf.
(Domain Specific Languages)

Rebol enthält für Dialekte eine komplette „Parsing Engine“. Mit verschiedensten Daten zu arbeiten wird zum Kinderspiel.

PARSE ist gut für „Pattern Matching“ geeignet.

```
length? second parse join number ".." "."

>> number: 1234.12
>> join number “..”
== “1234.12..”
>> parse “1234.12..” “.”
== ["1234" "12" ""]
>> second ["1234" "12" ""]
== “12“
>>length? “12“
== 2
```

Eigene Sprachen sind auch möglich.

```
parse [x 2 x “REBOL"] [
    some [
        'x
    set value [ integer! | string! ]    (
            print either integer? value
                [ value * 2 ]
                [ value ]
        )
    ]
]
```

```
parse [2 @ 3] grammar
** Syntax Error: Invalid email -- @

parse [2 test@test.com 3] grammar
```

Dialekte sind ein Dialekt des „Data Exchange Dialects“, also nicht ganz frei, da alle Regeln für Rebol Code auch für den Dialekt gelten.

SAPHIRION

Alle notwendigen Funktionen für Netzwerke sind direkt integriert.

Alle wichtigen Protokolle werden unterstützt.

Protokoll	Zweck
DNS	Domain Name Service
Finger	Information über User ausdessenUserprofil
Whois	Information über Domain Registration
Daytime	Network Time Protocol
HTTP	Hypertext Transfer Protocol
SMTP	Simple Mail Transfer Protocol
POP/IMAP	Post Office Protocol / Internet Message Access Protocol
FTP	File Transfer Protocol
NNTP	Network News Transfer Protocol
TCP	Transmission Control Protocol
UDP	User Datagram Protocol

Protokolle sind eine Teilmenge von „Schemes“

„Schemes“ nutzen Standard URI Notation.

Netzwerk

Alle notwendigen Funktionen für Netzwerke sind direkt integriert.

Alle wichtigen Protokolle werden unterstützt.

1. Öffne ein GUI
2. Lese den Inhalt einer Web-Adresse
3. Sende den Inhalt per Email

```
view layout [
u: fielduser@rebol.com
h: fieldhttp://
btn"Send" [
        send to-email u/text read to-urlh/text
        alert "Sent”
]
]
```

Print Web-Server ID

```
p: open http://www.rebol.com:80
print p/locals/headers/server
close p
```

- Full-Featured Apache-Class Web Server
- 500 simultaneous requests
- REBOL server pages support
- 390 KB (Source Code, Dokumentation, Bilder, Beispiele)
- Windows, Linux, OSX

REBOL / Service

- Lightweight Network Services (SOA)
- Built-in Standard Service Model
- Lightweight Implementation (Server 65KB, Client 40KB)
- Secure by Default (AES, DES, RSA, Blowfish)
- Transport Independent (TCP, HTTP, BEEP)
- Multi-Service Command Interfaces

Beispielanwendungen

REBOL / Service

SOA

out-of-the-box

Synchroner Aufruf:

>>do-service tcp://server:8000 [time]

== 17-May-2009/11:38:47+2:00

Asynchroner Aufruf:

>>send-service/action tcp://server:8000 [time] [printresult]

Asynchroner Aufruf und warten auf Ergebnis:

req: send-service tcp://server:8000 [time]

<do-some-other-stuff>

result: wait-servicereq

printresult

Better, denn Rebol mixt verschiedene Ansätze mit vielen Datentypen

und das steigert die Produktivität.

BETTER

Klassischer Ansatz:

```
>>team: [Programmer "Ginger Rogers" Tester none]
>>pointer: find team 'Tester
>>pointer: nextpointer
>>changepointer "Judy Garland“
== team: [Programmer "Ginger Rogers" Tester „Judy Garland“]
```

Rebolish Version:

```
>>changenext findteam'Tester "Judy Garland“
```

Es geht noch kürzer: team/tester: „Judy Garland“

In Rebol schreibt man A B C anstatt A(B(C))
und kann damit schöne Sätze bilden.

SAPHIRION

action!	datatype-relative native function (polymorphic)	money!	arbitrary precision decimal numbers (denominations)
binary!	string series of bytes	native!	direct CPU evaluated function
bitset!	set of bit flags	none!	no value represented
block!	series of values	object!	context of names with values
char!	8bit and 16bit character	op!	infix operator (special evaluation exception)
closure!	function with persistent local values (indefinite)	pair!	two dimensional point or size
datatype!	datatype value	paren!	automatically evaluating block
date!	day, month, year, time of day, and timezone	path!	refinement path for functions, objects, files, etc.
decimal!	64bit floating point number (IEEE standard)	percent!	special form of decimals (used mainly for layout)
email!	email address	port!	external series, an I/O channel
end!	internal marker for end of block	rebcode!	virtual machine function
error!	error value	refinement!	variation of meaning or location
event!	user interface event (efficiently sized)	routine!	external library function
file!	file name or path	set-path!	definition of a path's value
frame!	internal context frame	set-word!	definition of a word's value
function!	interpreted function (user-defined or mezzanine)	string!	string series of characters
get-path!	the value of a path	struct!	native structure definition
get-word!	the value of a word (variable)	tag!	markup string (HTML or XML)
gob!	graphical object	task!	evaluation environment
handle!	arbitrary internal object or value	time!	time of day or duration
image!	RGB image with alpha channel	tuple!	integers sequence (colors, versions, IP addresses)
integer!	64 bit integer	typeset!	set of datatypes
issue!	identifying string or script marker	unicode!	string of unicoded characters
library!	external library reference	unset!	no value returned or set
lit-path!	literal path value	url!	uniform resource locator or identifier
lit-word!	literal word value	utype!	user defined datatype
logic!	boolean true or false	vector!	multideminsional directly mapped arrays (of same datatype)
map!	hashed name-value pairs	word!	word (symbol or variable)
module!	loadable context of code and data		

In Rebol (R3) sind 58 Datentypen
mit passender Semantik verfügbar.

SAPHIRION

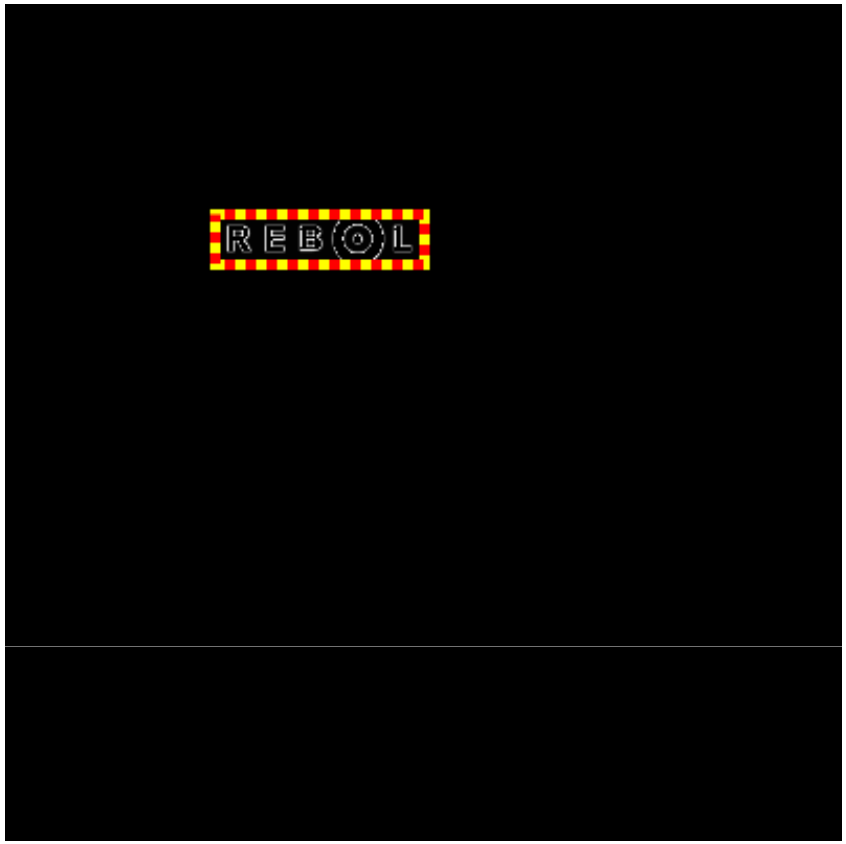
image!


```
>> image logo.gif
```

image!

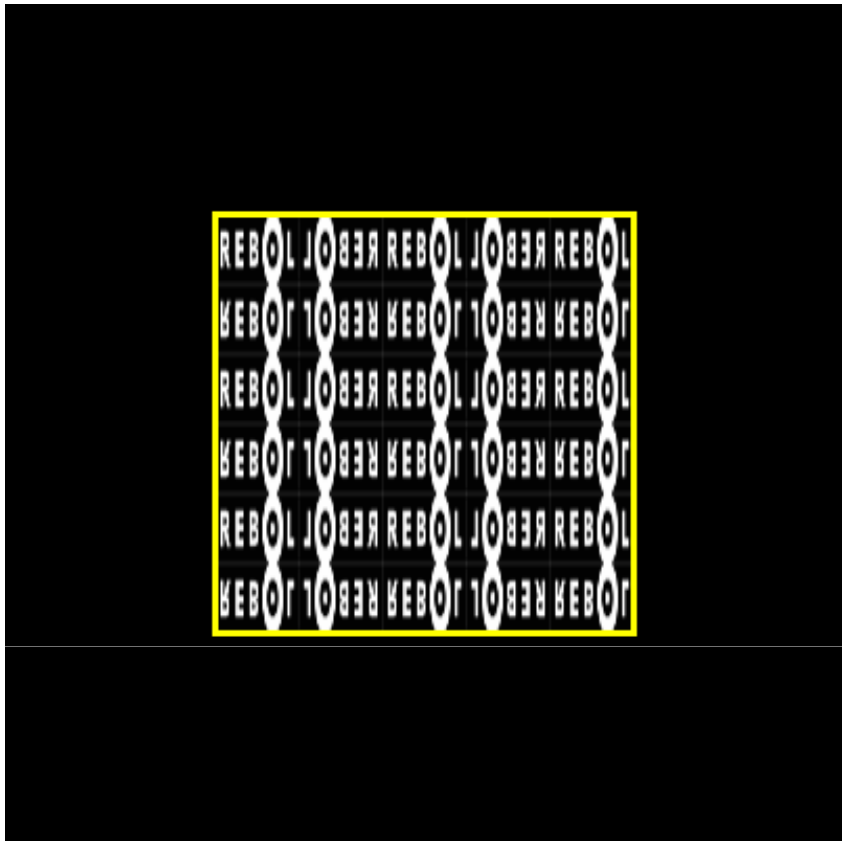

```
>> image logo.gif 100x100 300x200
```

image!




```
>> pen yellow line-width 5 line-pattern red 5 5
>> image logo.gif 100x100 254.254.254 border
```

image!



```
>> line-width 3
>> pen yellow
>> image logo.gif 100x100 300x300 border reflect 0 0 500
    144
```

image!

>> image logo.gif 50x100 400x00 400x400 50x200
>> image logo.gif 10x10 350x200 250x300 50x300

closure!

WORDs sind lokal “binded” für die Zeit der Auführung:

```
>>add2: func [cd] [[c + d]]
>> do add2 1 2
** Script error: c word is not bound to a context
```

Eine CLOSURE behält WORD VALUEs nach der Ausführung:

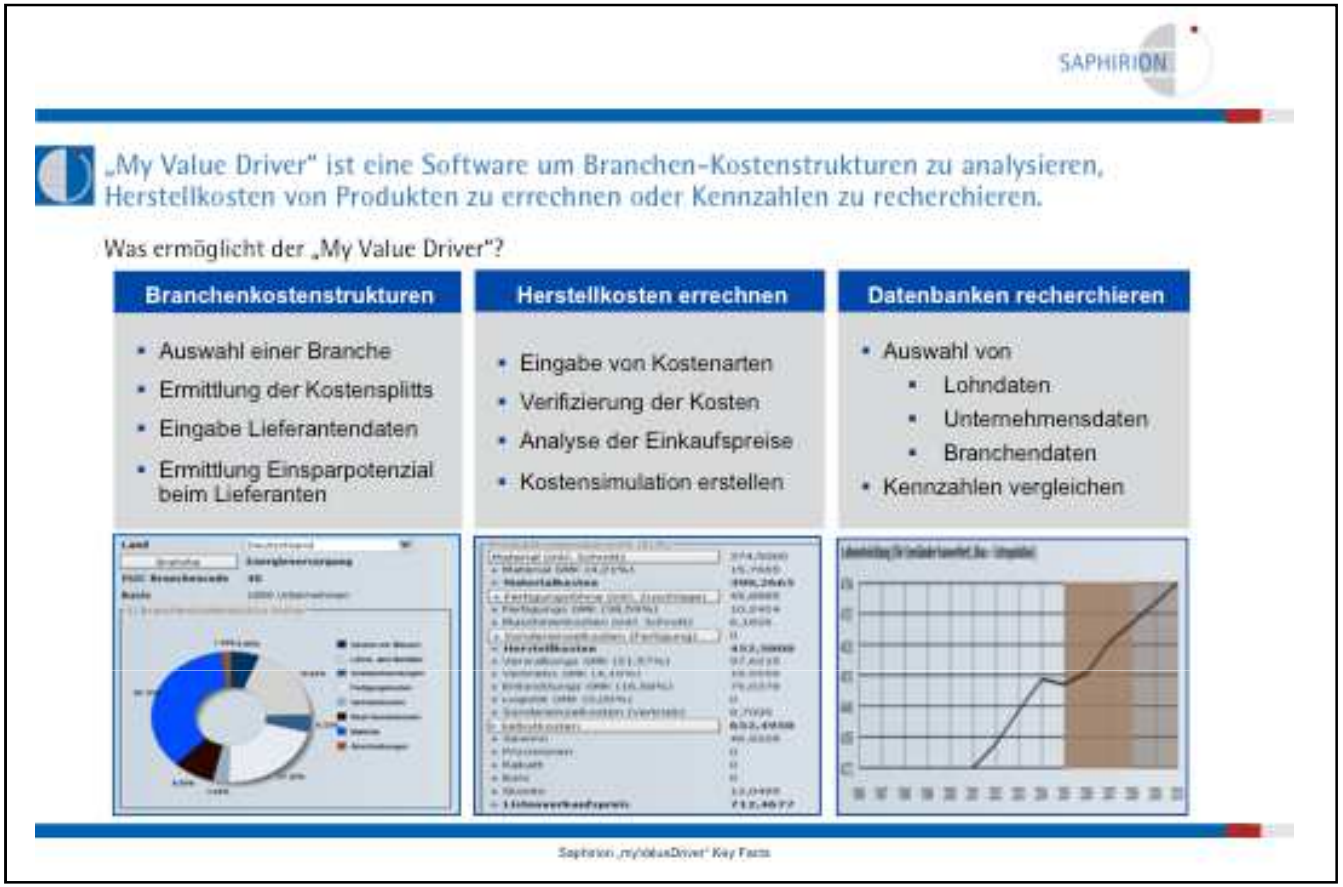
```
>> add2: closure [cd] [[c + d]]
>> do add2 1 2
== 3


>> make-adder: closure[x] [func [y] [x + y]]
>> add-10: make-adder 10
>> add-10 5   == 15
```

CLOSURE ist ein spezieller Funktionstyp mit persistenten WORD / VALUES

Rebol ist extrem Produktiv:

Kleine Scripte, große Wirkung

Sehr gute Community.



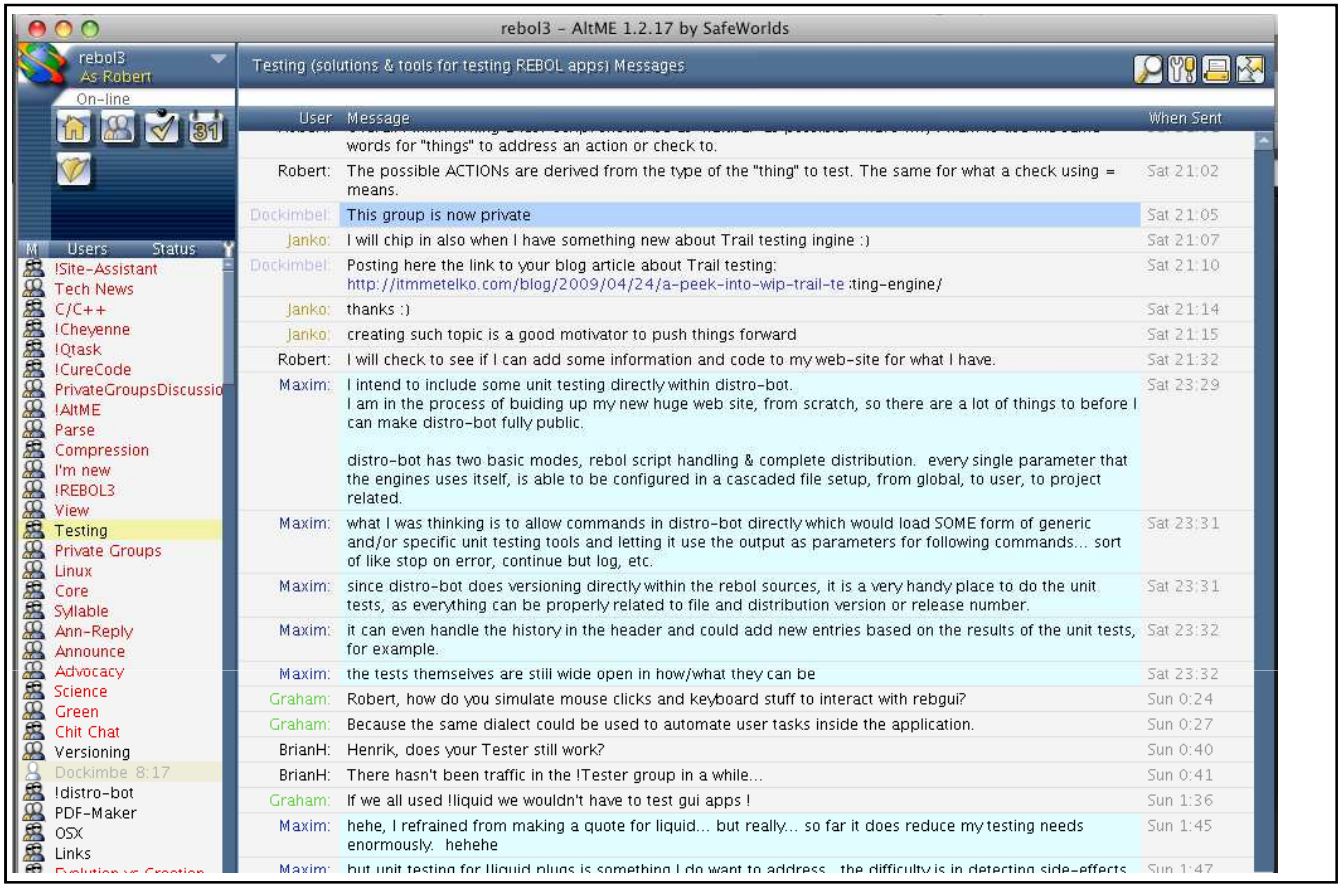
- 1,5 MB Source-Code, 1 MB EXE
- 7 Monate Entwicklungszeit für V1.0
- 2 FTE Entwickler
- Rebol Script + DLL für Datenbank, COM Steuerung, Lizenzsierung

- Wettbewerb: VBA, 4 Jahre Entwicklungszeit (1 FTE Entwickler)

Rebol ist extrem Produktiv:

Kleine Scripte, große Wirkung

Sehr gute Community.



- DarkNet Messaging
- ca. 100 aktive Reboler
- „The Place to be“
- Account via Mailingliste

- Mailingliste
- R3/Beta Forum
- rebol.org Library
- Carl‘s BLOG

Mit R3 kommt:

„Programming in the Large“.

„REBOL 3 is perfect for lightweight distributed applications”

Continued Lightweightness: The footprint will still fall under 1 MB.

Programming-in-the-large: Adding modules (namespaces and scoping control), multi-tasking, more-efficient GC, and more debugging "hooks and tools".

Hybrid Open Model: Redesigning the REBOL kernel to allow a greater degree of open interfaces, both internal (built-in) and external (plug-ins).

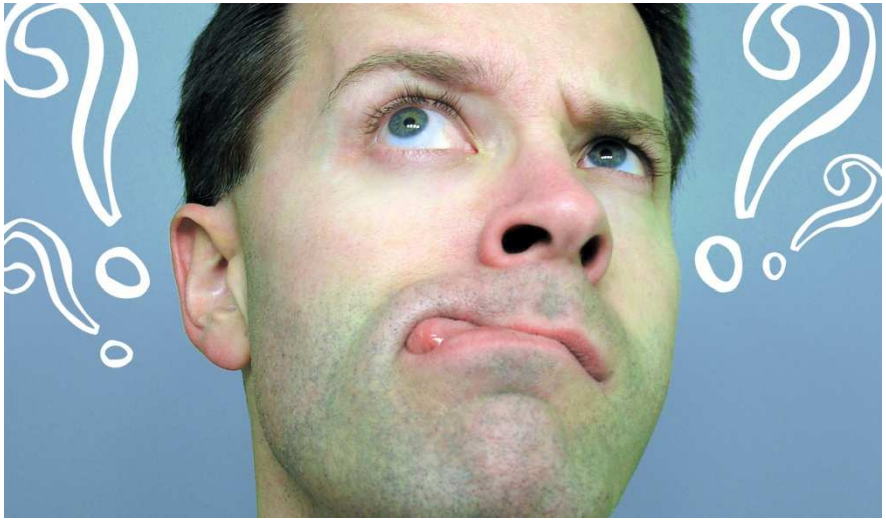
REBOL-in-the-Browser: Providing the necessary source code (as an open interface) to get your help and make REBOL work across all browsers on all operating systems.

REBOL/Services Built-In: Securely connect to other REBOL nodes and services and efficiently exchange information.

Optimized Graphics , Greater Locality Support (UTF-8), Resident Storage Method

Rebol:

Smarter
Faster
Better



Vielen Dank!