What is Red?
Why Red?

Let's start by taking a bird's eye view.
Why Red?

We are still at a very early age of computing…
Why Red?

…and in the future, AI will laugh at our technologies!
Why Red?

There are thousands of programming tools/languages...

- A\# .NET
- A\# (Axiom)
- A-0 System
- A+
- A++
- ABAP
- ABC
- ABC ALGOL
- ABLE
- ABSET
- ABSYS
- Abundance
- ACC
- Accent
- Ace DASL
- ACT-III
- Action!
- ActionScript
- Ada
- Adenine
- Agda
- Agilent VEE
- Agora
- AIMMS
- Alef
- ALF
- ALGOL 58
- ALGOL 60
- ALGOL 68
- Alice
- Alma-0
- AmbientTalk
- Amiga E
- AMOS
- AMPL
- APL
- AppleScript
- Arc
- ARexx
- Argus
- AspectJ
- Assembly language
- ATS
- Ateji PX
- AutoHotkey
- Autocoder
- AutoIt
- AutoLISP / Visual LISP
- Averest
- AWK
- Axum
Why Red?

…but we are still searching for the right one.
Why Red?

We are often wasting time to workaround dead ends…

- Exploding complexity
- Slow performances
- Bloatware
- Black boxes
Why Red?

…while we should have fun working on computers!
Why Red?

Our computing world is going through deep changes…

"Massively parallel 64 core computer costs $99" - Parallela

We are moving away from traditional computers.
Why Red?

...yet we are using rusty software solutions.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>=</td>
<td>C</td>
<td>17.009%</td>
<td>+0.00%</td>
<td>A</td>
<td>1972</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>=</td>
<td>Java</td>
<td>16.655%</td>
<td>+0.39%</td>
<td>A</td>
<td>1995</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>↑</td>
<td>Objective-C</td>
<td>10.355%</td>
<td>+1.26%</td>
<td>A</td>
<td>1983</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>↓</td>
<td>C++</td>
<td>8.810%</td>
<td>-0.54%</td>
<td>A</td>
<td>1983</td>
</tr>
<tr>
<td>5</td>
<td>7</td>
<td>↑↑</td>
<td>PHP</td>
<td>5.997%</td>
<td>+0.70%</td>
<td>A</td>
<td>1995</td>
</tr>
</tbody>
</table>
Why Red?

*Rebol was partially answering those concerns…*
Why Red?

…doing wonders, only recently acknowledged,…
Why Red?

…but Red will go far beyond!
What is Red?

• A "full stack" programming language
  – True general-purpose programming solution
  – A stand-alone toolchain

• A tool for empowering users
  – Simple to use
  – Unlimited usage, no arbitrary restrictions
  – Feeling "in control" again

Red is bringing back the fun to programming!
Natural scope of application

Abstraction level

Meta DSL

DSL

Scripting

Applications

OS

Drivers

Hardware

C
Pascal
Java
C++

Python
Ruby
Javascript
REBOL
Rascal
Red
Red/System

ASM
A standalone toolchain 1/5

Embedded
- Android devices
- iOS devices
- Raspberry Pi
- Arduino boards
  (AVR 8/32-bit)

Desktop
- Windows
- Linux
- MacOS X
- Syllable
- FreeBSD
- .NET
- JVM
- Javascript

Virtual Machines

Red
A standalone toolchain 2/5

- Compile script and run it from memory
  $ red script.red
- Compile script and output an executable
  $ red -o script script.red
- Cross-compile script and output an executable
  $ red -t Windows -o script script.red
- Compile a Red/System script and output an executable
  $ red script.reds
- Compile script as shared library
  $ red -dlib script.red
- Launch Red in REPL mode
  $ red

Currently: >> do/args %red.r "-dlib script.red"
A standalone toolchain 3/5

- Cross-compilation made right!

```
$ red -t <TargetID> script.red
```
A standalone toolchain 4/5

Red toolchain

Compiler

x86, ARMv5+, ARMv7, Thumb, x86-64,…
JS, JVM bytecode, Dex, MSIL,…

Linker

PE, ELF, Mach-o, Intel-HEX
Executable, Shared library
Static library, kernel driver,…

Packager

Android - APK
iOS - IPA
Webapps – WAR
…
### A standalone toolchain 5/5

<table>
<thead>
<tr>
<th>2011-2013</th>
<th>2014</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bootstrapped</td>
<td>Self-hosted</td>
</tr>
</tbody>
</table>

- **in Rebol 2** | **in Red**

*JIT-Compiler*

*Last dependency to remove for full feature access!*
Red language 1/4

• Syntax and semantics very close to Rebol
  – Definitional scoping
  – Dynamic binding

• Paradigm-neutral
  – Imperative, OOP, functional, symbolic,…

• Optionally typed arguments and locals
  – From pure interpretation to very specialized compilation
  – Type inference by default, when possible
Red language 2/4

• Unicode support
  – Input sources in UTF-8, external codec for other encodings
  – Auto-adaptative internal representation: UCS-1, UCS-2 or UCS-4

• Concurrency support
  – Task parallelism: light threads over multiple cores with Actors.

• Meta-DSL abilities
  – DSL-maker dialect (higher-level than PARSE)
Red language 3/4

- Compiles to Red/System code

\[ a: 1 + b \]

**Red input**

```
Red input
```

**Red stack**

```
Red stack
```

**Red/System output**

```
Red/System output
```

```
stack/mark-native ~set
word/push ~a
stack/mark-native ~add
integer/push 1
word/get ~b
actions/add*
stack/unwind
word/set
stack/unwind
```
• Compiled, interpreted, JIT-compiled
Red code example

Red [title: "Hello"]

print "Hello World!"

inc: func [n [integer!]][n + 1]

foo: function [a [integer!] /bar return: [string!]][
  z: "zero"
  either integer? a [
    return z
  ][
    append "result=" mold inc a
  ]
]
Red/System overview

- Purely imperative, C-level language, with a Red syntax
- Statically compiled, x4 slower than C (naïve compilation for now)
- Namespaces support: context, with

- Limited type system:
  - Logic!, byte!, integer!, float!, float32!, struct!, pointer!, c-string!, function!
  - Simple type inference
  - Type casting supported
  - Values type reflexion (variadic RTTI functions)

- Compiler directives: #define, #include, #import, #syscall, #if, #either, #switch,…

- Low-level CPU support (interruptions, I/O, stack, privileged mode)
- Inlined ASM support
Red project metrics

- BSD license (BSL for the runtime parts)

- Source code hosted on Github since March 2011
  - version 0.3.2, 9 committers, ~2050 public commits
  - 496 tickets in bugtracker (95% closed)
  - ~18000 unit tests (framework built by Peter WA Wood)

  - 390 KiB of sources for Red/System
  - 8500 LOC for Red/System compiler
  - 2600 LOC for Red/System linker

  - 2800 LOC for Red compiler
  - 800 LOC for Red interpreter
  - 14000 LOC for Red runtime library (~150KiB compiled on x86)
Red remaining tasks for 1.0, until…

- Complete core parts:
  - implement object!, error!, typeset!, binary!, decimal!, date!, time!, …
  - add proper error handling and arguments type-checking
  - implement I/O infrastructure
  - implement concurrency support
  - define a module system and modular compilation

- Provide a minimal Red IDE

- Documentation:
  - Write Red reference documentation
  - Write some first-steps tutorials

- Prepare a new red-lang.org site for 1.0 launch
… the real Red!

• Real Red will be the 2.0
… the real Red!

• Final compiler architecture
  – Two-stage JIT-compiler, with optional AOT optimizations
  – Documented public API for lexer, parser, emitter and linker
  – Plugin-oriented internal architecture
  – Written purely in Red (self-hosted)
  – Red/System: improved and cleaned-up syntax and semantics
  – Very open to contributors!

• Implications for current codebase in Rebol 2
  – Disposable code with limited lifetime
  – Not documented, monolithic, rigid, not meant for contributions
Project Organisation

- 2 collaborators on Github
- 11 contributors
- 2 mailing-list admins
- 3 Facebook Red page admins
- iiqux (IRCbot for reporting commits)
- You?
Project Funding
Thank you for listening.

Any questions?