## Language Geek 2.0

Remix D

3 Goals

## Share Interesting Tidbits of Languages I Enjoy </1>

## Show That Language Effects How You Think

# Inspire At Least One Of You to Learn A New Language 

## Have you ever noticed?

## Some Languages

## ( Spanish )

## Sound Fast

## While Others

( Mandarin )

## Sound Slow?

## Study

English, French, German, Italian, Japanese, Mandarin, Spanish, and Vietnamese (baseline)

## read 20 different texts

Native language

## Crunch the numbers

2 data points for each

# (1) average information density 

## / <br> syllable

## (2) average syllables / second

Mandarin density: 0.94 rate: 5.18

# Mandarin density: 0.94 rate: 5.18 

Japanese density: 0.49
rate: 7.84

# Mandarin English Japanese density: 0.940 .91 density: 0.49 rate: $5.18 \quad 6.19$ rate: 7.84 

## Something Fascinating

## Work through the math

All languages

## Same Information Density

/
Second

## Clearly

## Shows Threshold of Understanding

## 40,000+ Years Of Work

Written?

Roughly 5000

## Something Like That

# Mesopotamia? 

## Egypt?

Pakistan?

## Does written language follow the same rules?

No

## Information Density

( and our ability to understand )

## Varies SIGNIFICANTLY by Language

## However

## Written Languages

## Have Differences

## Have "Features"

## Spanish

# Consistent Rules Few Grammatical Exceptions 

## Easier to Learn

Mandarin

## Symbolic System

## Little Drift Over Centuries

(Have you read Chaucer!?)

## Seek the Perfect Language

## Esperanto

## Volapük

## Interlingua

Many More

## Fast Forward >>

## Programming Languages

## Different Languages

## Different Advantages

## So

## Let's Look

## What Makes Languages Awesome?

C
\#define REG(a) *(volatile uint32_t*)(a) \#define PIO_COR REG(0x5000)

## C is incredibly low-level

## Free Access To Everything

## "Portable Assembly"

## Good for Manual Optimization

## What does this do?

$$
\begin{aligned}
& x=x \wedge y \\
& y=x \wedge y \\
& x=x \wedge y
\end{aligned}
$$

## (It swaps two words. No extra memory)

Efficient!

Not Always Readable

## Rarely Quick to Write

## You Specify EVERYTHING

## What CAN you develop in quickly?

## Ruby

## Primarily a Scripting Language

## "Principle of Least Surprise"

## "Duck Typing"

## Biggest Goal

## "Make Programming Fun"

a = [1, 2, 3, 4, "fish", "stix", 7] puts a[0], a[-1], a[4..-1].join(' '), a[8]
a = [1, 2, 3, 4, "fish", "stix", 7] puts a[0], a[-1], a[4..-1].join(' '), a[8]

1
7
fish stix 7 nil

## Flexible Containers

## Flexible Array Handling (nils, negative indexing)

## Powerful String Handling

## Ruby is Glue

## Because, who wants to write this in $C$ :

## Because, who wants to write this in C: Open a TCP/IP socket and read 50,000 bytes.

Because, who wants to write this in C: Open a TCP/IP socket and read 50,000 bytes. Force the incoming text to have unix line endings

Because, who wants to write this in C: Open a TCP/IP socket and read 50,000 bytes. Force the incoming text to have unix line endings, then write it to a file.

Because, who wants to write this in C: Open a TCP/IP socket and read 50,000 bytes. Force the incoming text to have unix line endings, then write it to a file. If there are any problems, make sure you close the port properly

Because, who wants to write this in C: Open a TCP/IP socket and read 50,000 bytes. Force the incoming text to have unix line endings, then write it to a file. If there are any problems, make sure you close the port properly and make sure you close that file handle too.

Because, who wants to write this in C: Open a TCP/IP socket and read 50,000 bytes. Force the incoming text to have unix line endings, then write it to a file. If there are any problems, make sure you close the port properly and make sure you close that file handle too. Also, watch for memory leaks.

## This is What Ruby is For

File.open("log.txt",'w') $\{|\mathrm{f}| \mathrm{f} \ll$
TCPSocket.new("192.1.168.1",3000).recv(50000).gsub(//r\n/,"\n")\}

File.open("log.txt",'w') do |f| $\mathrm{f} \ll$ TCPSocket.new("192.1.168.1",3000) .recv(50_000) .gsub(/\r\n/,"\n")
end

Make You Want to Cry?

## One More Awesome Thing

## Everything in Ruby is Dynamic

## Want to print ints as Roman Numerals?

## Just change FixedNum's .to_s method

## How about creating functions from text files?

class_eval do
Dir['scripts/*.meh'].each do |s| define_method("run_\# \{s\}")
\{ eval(s) \}
end
end

## In Ruby, Line Between Data and Code Blurred

## Which Leads Us To

## Self

## Pure Object Oriented

Prototype

With "Slots"

## Slots can be

data
methods
parent

## Start by Cloning

## Change As Needed

No Classes or Types

## Inheritance by Parent Slots

## Undefined Slots

## Search up Through Each Parent

## Until Match

## Very Flexible

## Can Set Slot As Data

Then Signal a Slot to Run

Another Unique Aspect

## Graphical and Text Interfaces

## Interchangeable

4an object
Module: balanoe 100 :
deposit: d
balance: balance + d 日
withdraw: w balance: (0 max: balane - w) 日

## Can Execute Directly From Diagrams

## Invented JIT Compilation

## Later Forked the Become Java

## MANY Languages Run on JVM

Clojure

## (Think LISP on JVM)

## (+ 1 1)

## (+ (* m x) b)

(class (/ 1 3))

## (class (/ 1 3)) clojure.lang.Ratio

# recursion! <br> (defn count [v] (if (empty? v) 0 (inc (count(rest v))))) 

## infinite sequences! (repeat "hi")

(take 3 (repeat "hi"))

## Whoa! Look at that again.

(take 3 (repeat "hi"))

## Lazy Evaluation!

## Pop quiz!

# (defn f [n] (apply * (take n (iterate inc 1)))) <br> what does this do? 

# (defn f [n] (apply * (take n (iterate inc 1)))) 

## it computes n !

if that doesn't feel foreign yet

## Prolog

not imperative
(it's not a recipe)
it's declarative
(declares facts and constraints)
likes (bryan, skating).
likes (kevin, skating).
likes (kevin, ruby).
likes (mark, ruby).
likes (mark, lua).
likes (dean, tcl).
hangout( $\mathrm{X}, \mathrm{Y}$ ) :- $-+(\mathrm{X}=\mathrm{Y})$, likes(X,Z), likes(Y,Z).
likes (bryan, skating).
likes (kevin, skating).
likes (kevin, ruby).
likes (mark, ruby).
likes (mark, lua).
likes (dean, tcl).
hangout( $\mathrm{X}, \mathrm{Y}$ ) :- $\+(\mathrm{X}=\mathrm{Y})$, likes( $\mathrm{X}, \mathrm{Z})$,
likes(Y,Z).
?- likes(bryan, skating).
yes
?- hangout(bryan, kevin).
yes
?- hangout(bryan, mark).
no
add some more rules:
category (language, ruby). category (language, lua). category (language, tcl). category (sport, skating).
likes_category(X,Y) :- likes(X,Z), category(Y,Z). (like an inner join in SQL)
?- likes_category(What, language).
What = kevin ;
What = mark ;
What = dean ;
pretty crazy huh?

## What about a language that is MEANT TO CRASH!?

## Erlang

## "Let it Crash"

## Hot-Swap Code

## built-in message-passing \& monitoring processes

loop() ->
process_flag(trap_exit, true), receive
new ->
register(process,
spawn_link(fun action:loop/0)), loop();
\{'EXIT', From, Reason\} -> self() ! new, loop()
end.

## I haven't even talked about

## I haven't even talked about

## Haskell <br> Digital Mars' D

## Scratch

Python Dylan I haven't even talked about Go Io

JavaScript
Lua
whew

## Clearly

There are MANY

## Fascinating Languages

## And There is a LOT to LEARN

## $<3>$ So Get Learning! </3>

</Language Geek="2.0">

