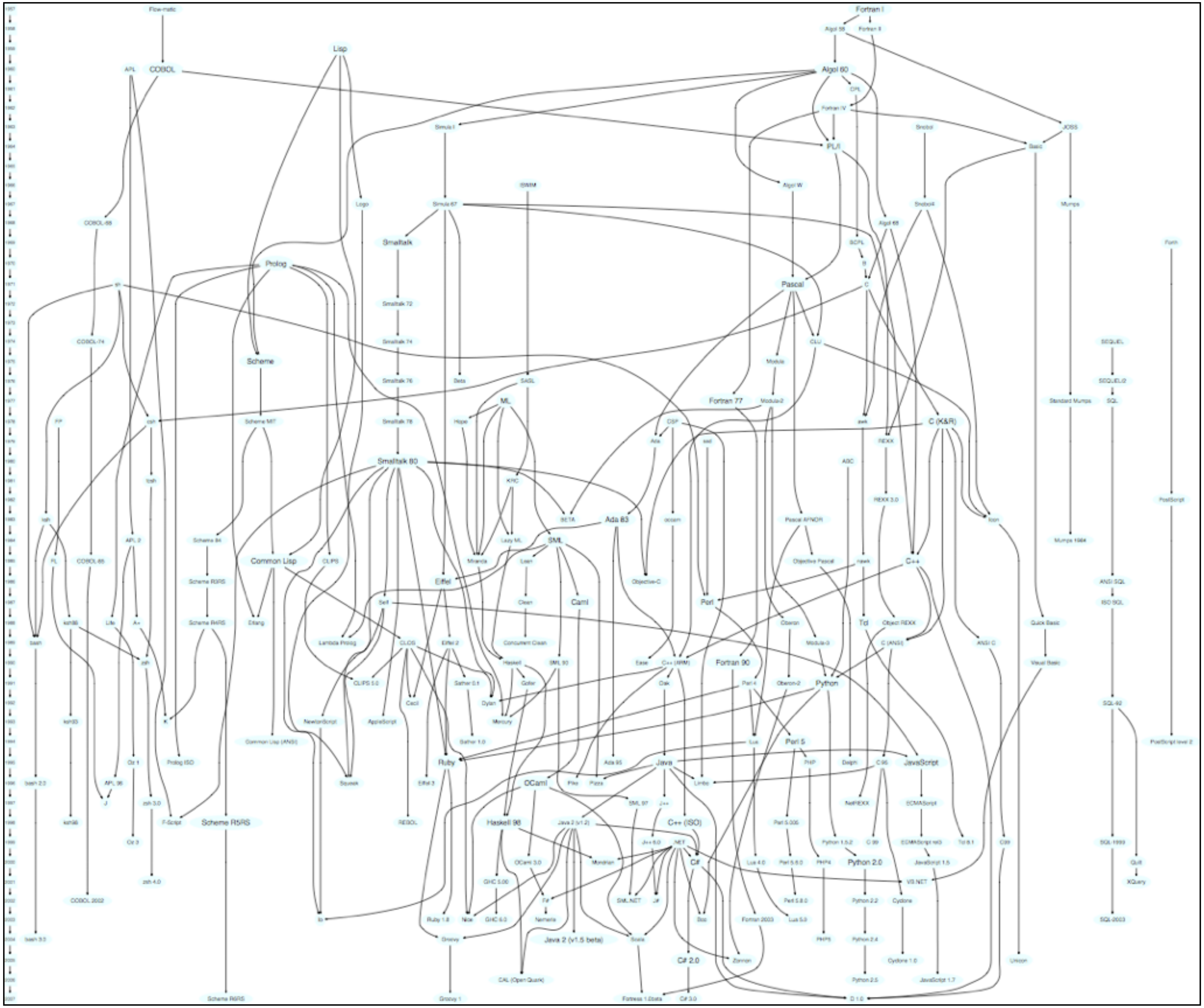
Topic List

Programming Languages:

- Family Trees.
- Characteristics.
- Typing Spectrum.

Family Tree



Flow-matic
COBOL
APL
Lisp
Fortran I
Algol 58
Fortran II
Algol 60
CPL
Fortran IV
Simula I
Snobol
JOSS
PL/I
Basic
Algol W
Logo
Simula 67
Snobol4
Mumps
COBOL-68
Algol 68
Smalltalk
BCPL
Forth
Prolog
B
C
sh
Pascal
Smalltalk 72
Smalltalk 74
COBOL-74
CLU
SEQUEL
Scheme
Modula
Smalltalk 76
Beta
SASL
SEQUEL/2
ML
Fortran 77
Modula-2
Standard Mumps
SQL
FP
csh
Scheme MIT
Smalltalk 78
Hope
CSP
awk
C (K&R)
Ada
sed
REXX
Smalltalk 80
ABC
tcsh
KRC
REXX 3.0
PostScript
ksh
BETA
Ada 83
occam
Pascal AFNOR
Icon
APL 2
Scheme 84
Lazy ML
SML
Mumps 1984
FL
COBOL-85
Common Lisp
CLIPS
Miranda
Lean
Objective Pascal
nawk
C++
Eiffel
Scheme R3RS
Objective-C
ANSI SQL
Self
Clean
Caml
Perl
ISO SQL
ksh88
Life
A+
Scheme R4RS
Erlang
Oberon
Tcl
Object REXX
Quick Basic
bash
Lambda Prolog
CLOS
Eiffel 2
Concurrent Clean
Modula-3
C (ANSI)
ANSI C
zsh
Haskell
SML 90
Ease
C++ (ARM)
Fortran 90
Visual Basic
CLIPS 5.0
Sather 0.1
Gofer
Oak
Perl 4
Oberon-2
Python
Cecil
Dylan
SQL-92
ksh93
K
NewtonScript
AppleScript
Mercury
Common Lisp (ANSI)
Sather 1.0
Lua
Perl 5
PostScript level 2
Oz 1
Prolog ISO
Ruby
Ada 95
Java
PHP
Delphi
C 95
JavaScript
bash 2.0
APL 96
Squeak
Eiffel 3
OCaml
Pike
Pizza
Limbo
J
zsh 3.0
SML 97
J++
NetREXX
ECMAScript
ksh98
F-Script
Scheme R5RS
REBOL
Haskell 98
Java 2 (v1.2)
C++ (ISO)
Perl 5.005
Oz 3
J++ 6.0
.NET
Python 1.5.2
C 99
ECMAScript ed3
Tcl 8.1
C99
SQL-1999
OCaml 3.0
Mondrian
C#
Lua 4.0
Perl 5.6.0
PHP4
Python 2.0
JavaScript 1.5
Quilt
zsh 4.0
GHC 5.00
VB.NET
XQuery
COBOL 2002
F#
SML.NET
J#
Perl 5.8.0
Python 2.2
Cyclone
Io
Ruby 1.8
Nice
GHC 6.0
Nemerle
Boo
Fortran 2003
Lua 5.0
SQL-2003
bash 3.0
Groovy
Java 2 (v1.5 beta)
Scala
PHP5
Python 2.4
C# 2.0
Zonnon
Cyclone 1.0
Unicon
CAL (Open Quark)
Python 2.5
JavaScript 1.7
Scheme R6RS
Groovy 1
Fortress 1.0beta
C# 3.0
D 1.0

Family Tree

Lisp - designed late 1950s - departure from existing languages (e.g. Fortran).

"Lisp is the Chuck Norris of programming languages."

Lisp, Fortran Cluster

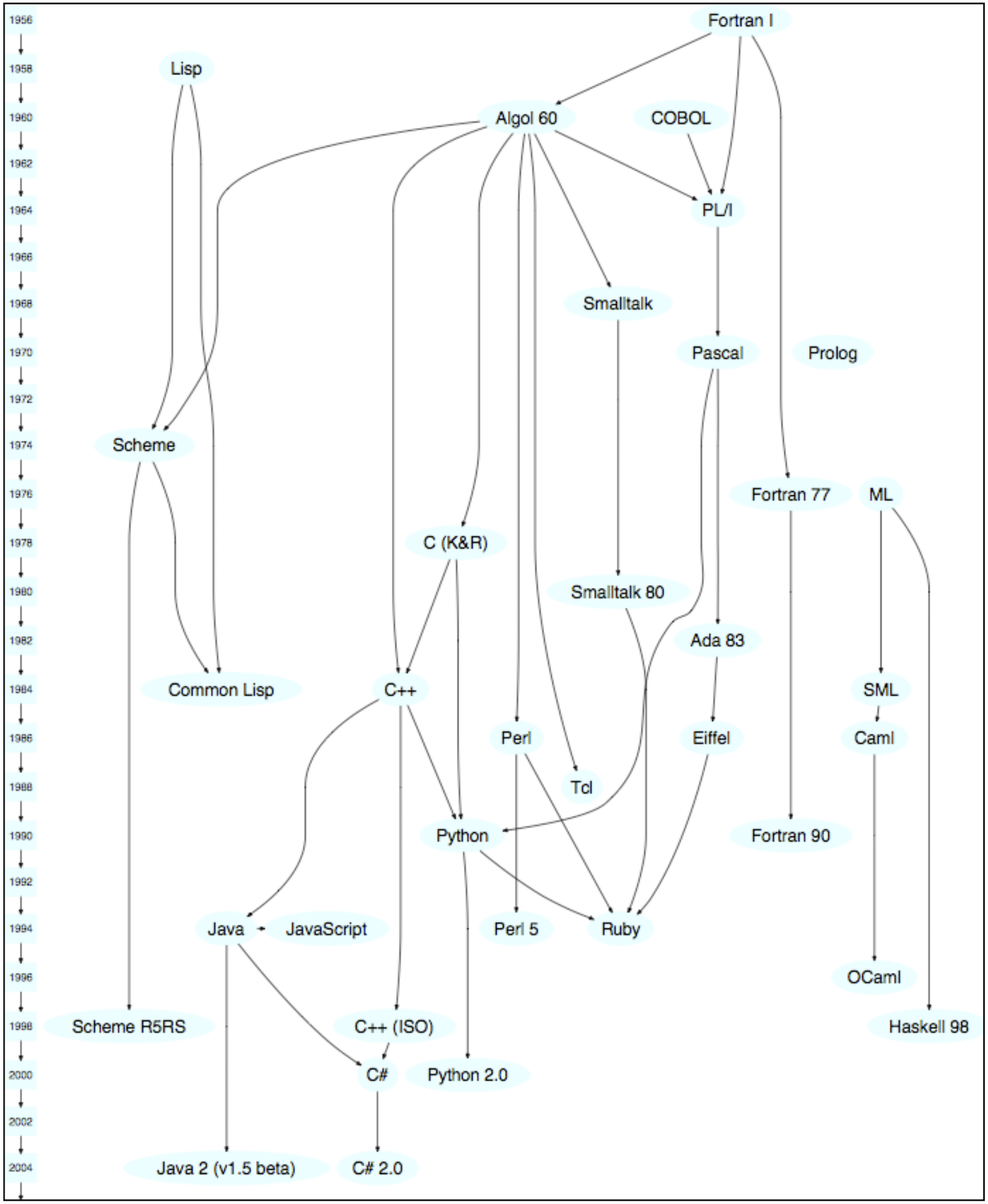
"Lisp is the bar other languages are measured against."

"Knowing Lisp demonstrates developer enlightenment."

https://hase85.wordpress.com/2013/01/15/best-of-lisp-quotes/

Family Tree

Lisp, Fortran Cluster



Family Tree

Smalltalk - pure OO language → everything is an object!

Objective C – borrows Smalltalk concepts & syntax.

Smalltalk Cluster

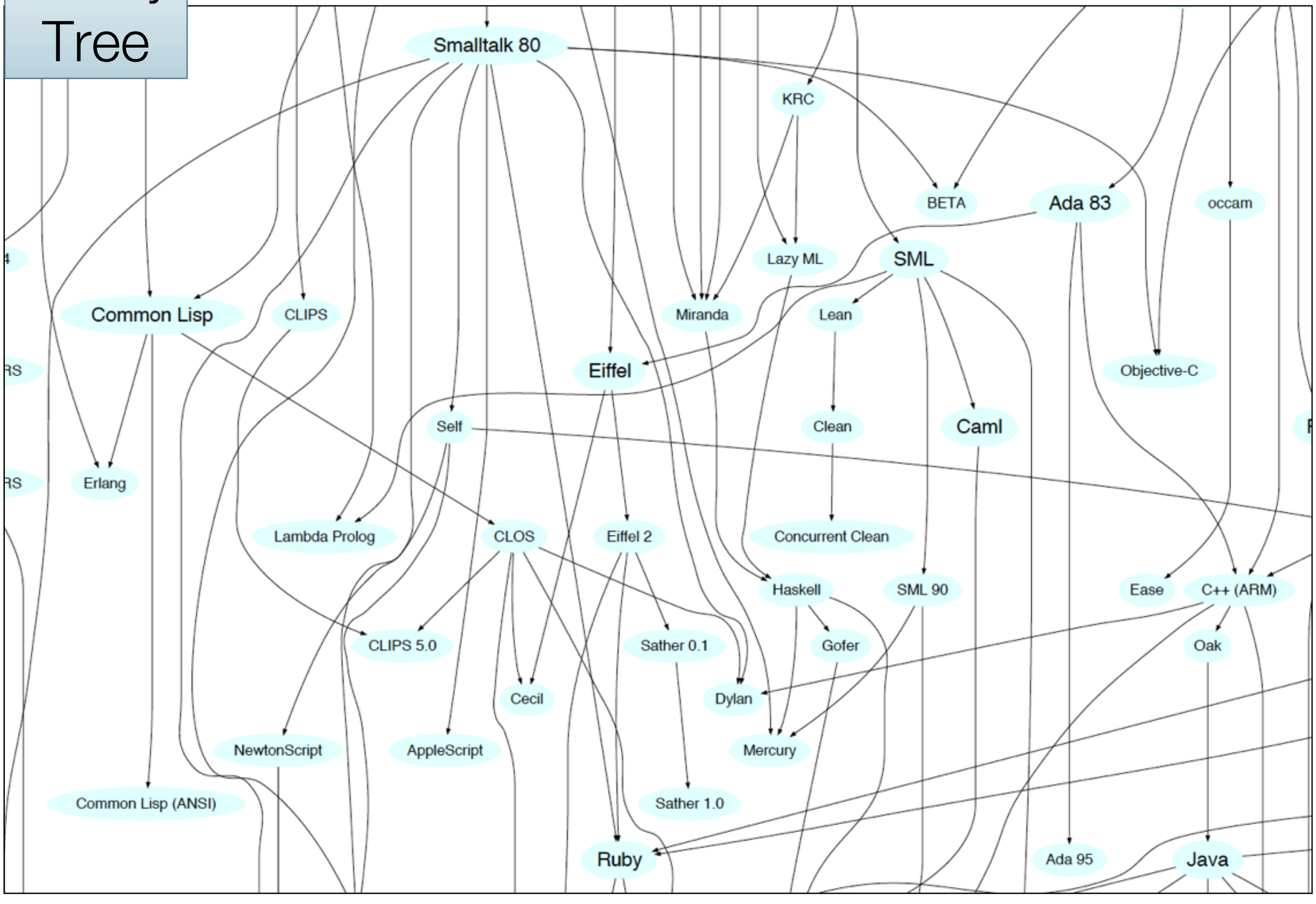
Java – borrows Smalltalk bytecode and garbage collection.

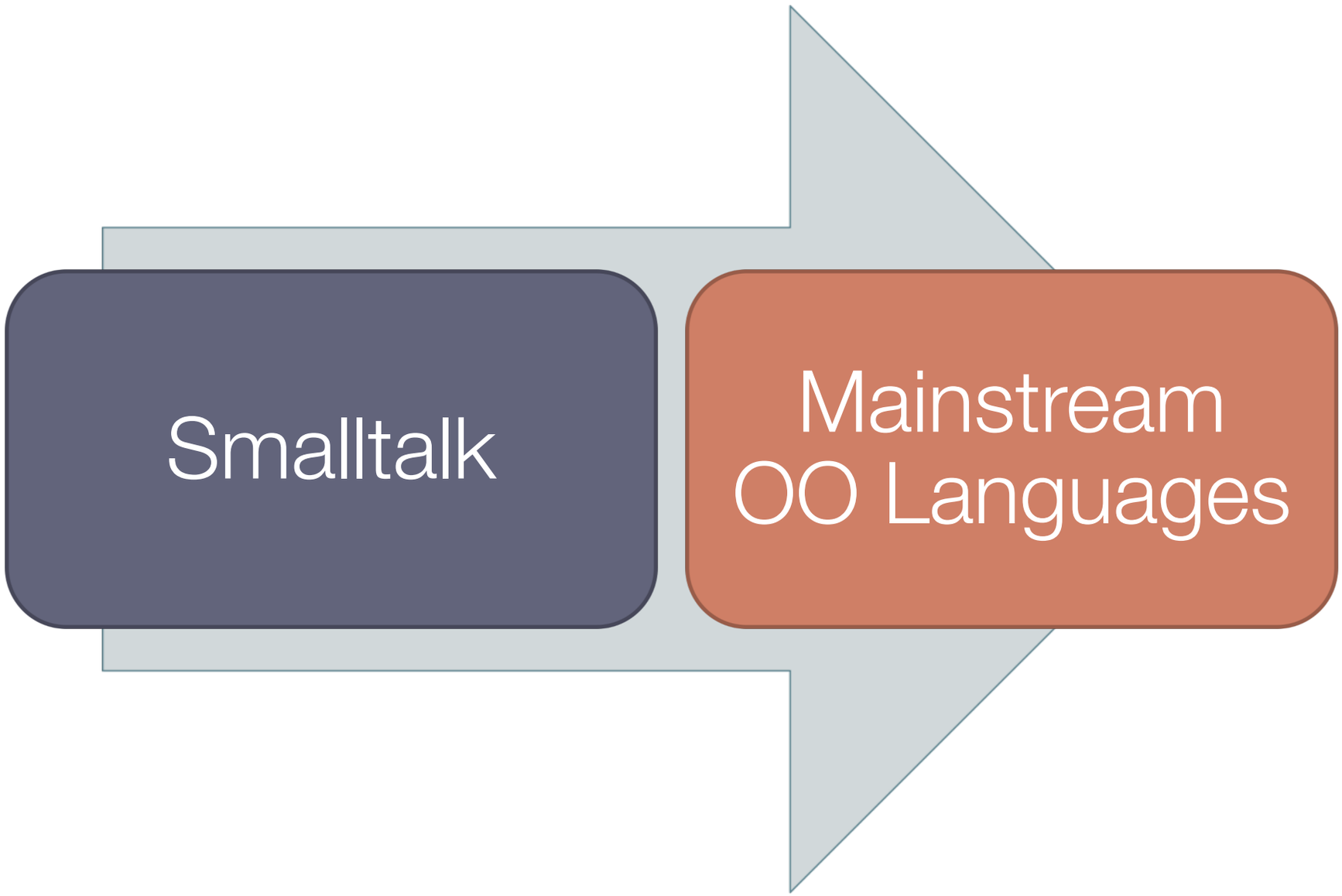
Ruby – pure OO model almost identical to Smalltalk.

https://www.cs.cmu.edu/~charlie/courses/15-214/2014-fall/slides/25-history-oo.pdf

Family Tree

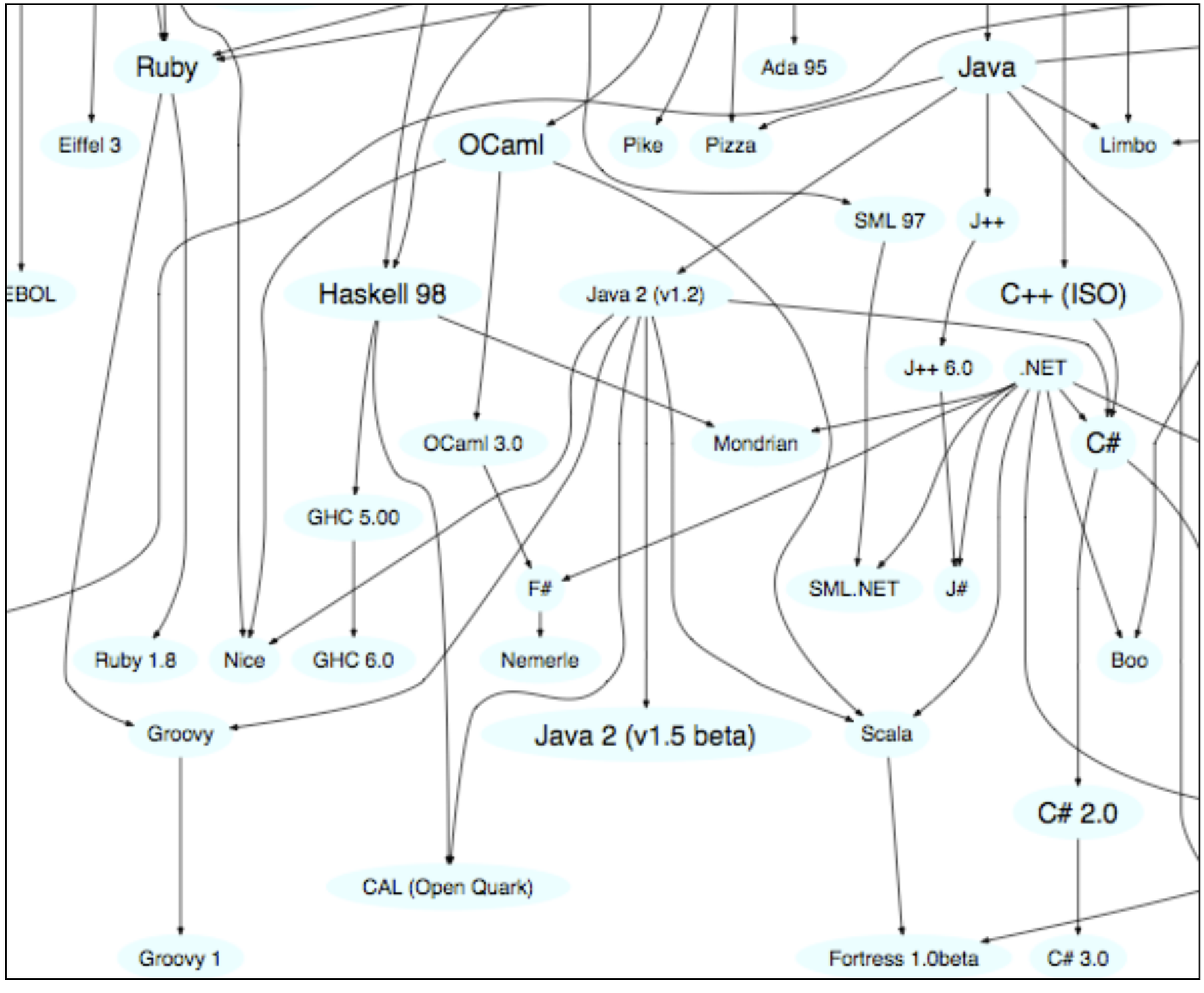
Smalltalk Cluster



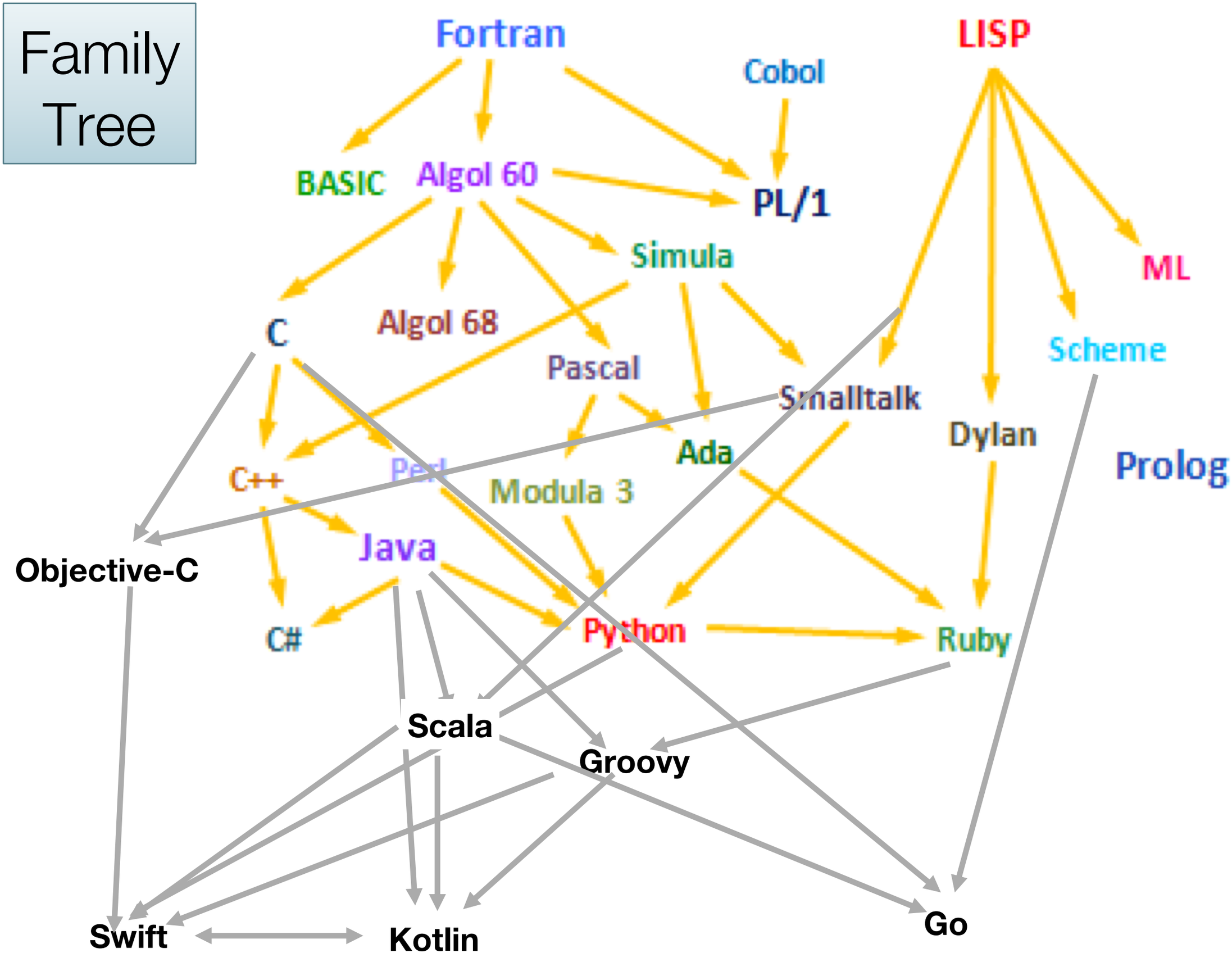

Smalltalk
Mainstream
OO Languages

Family Tree

Ruby, Groovy, Java, Scala Cluster



Family Tree


Fortran
LISP
Cobol
BASIC
Algol 60
PL/1
Simula
ML
C
Algol 68
Scheme
Pascal
Smalltalk
Dylan
Ada
Prolog
C++
Perl
Modula 3
Objective-C
Java
C#
Python
Ruby
Scala
Groovy
Swift
Kotlin
Go

Why so many programming languages?

Programming Languages as Reactions

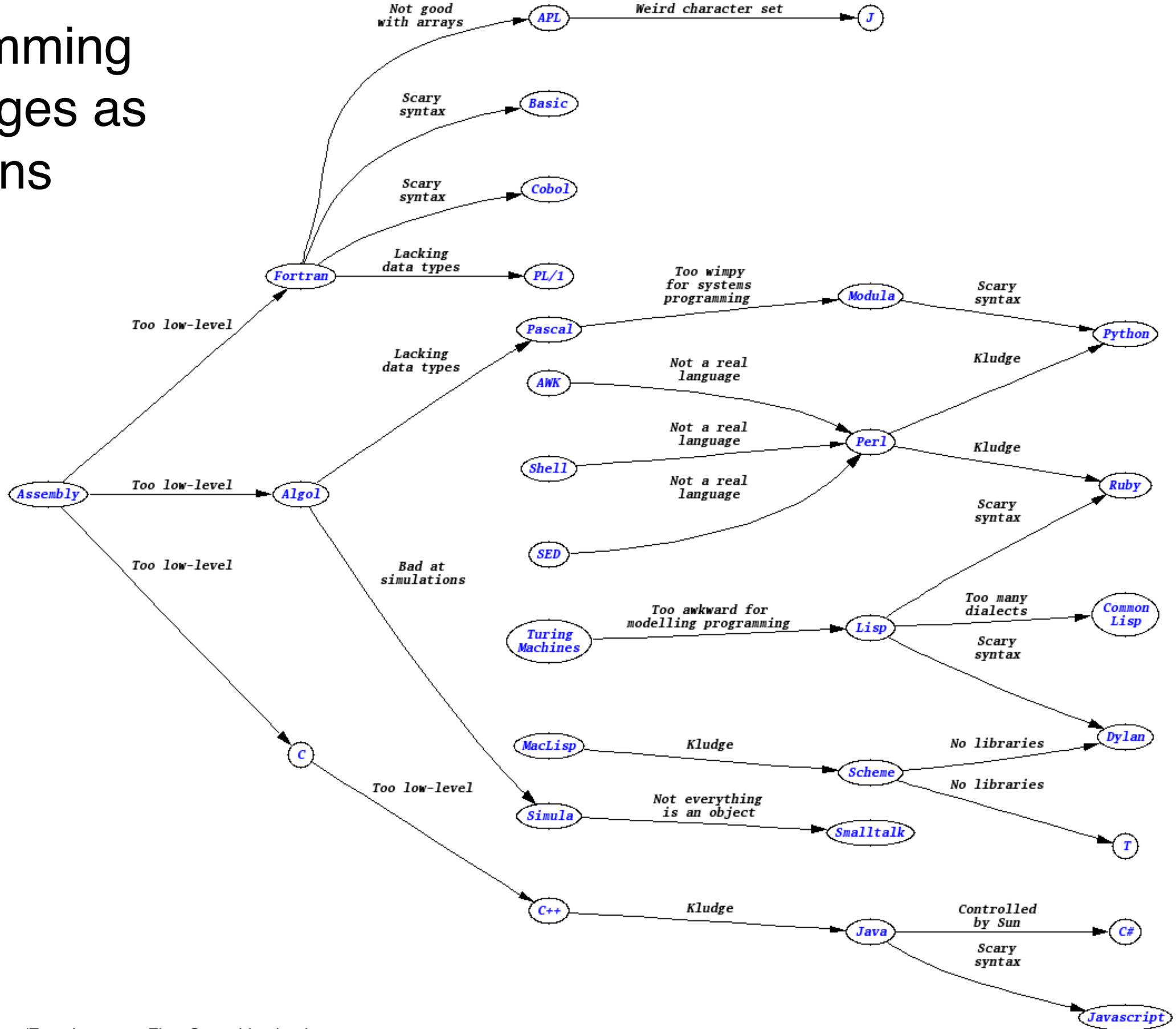
“Kevin Kelleher suggested an interesting way to compare programming languages:

to describe each in terms of the problem it fixes.

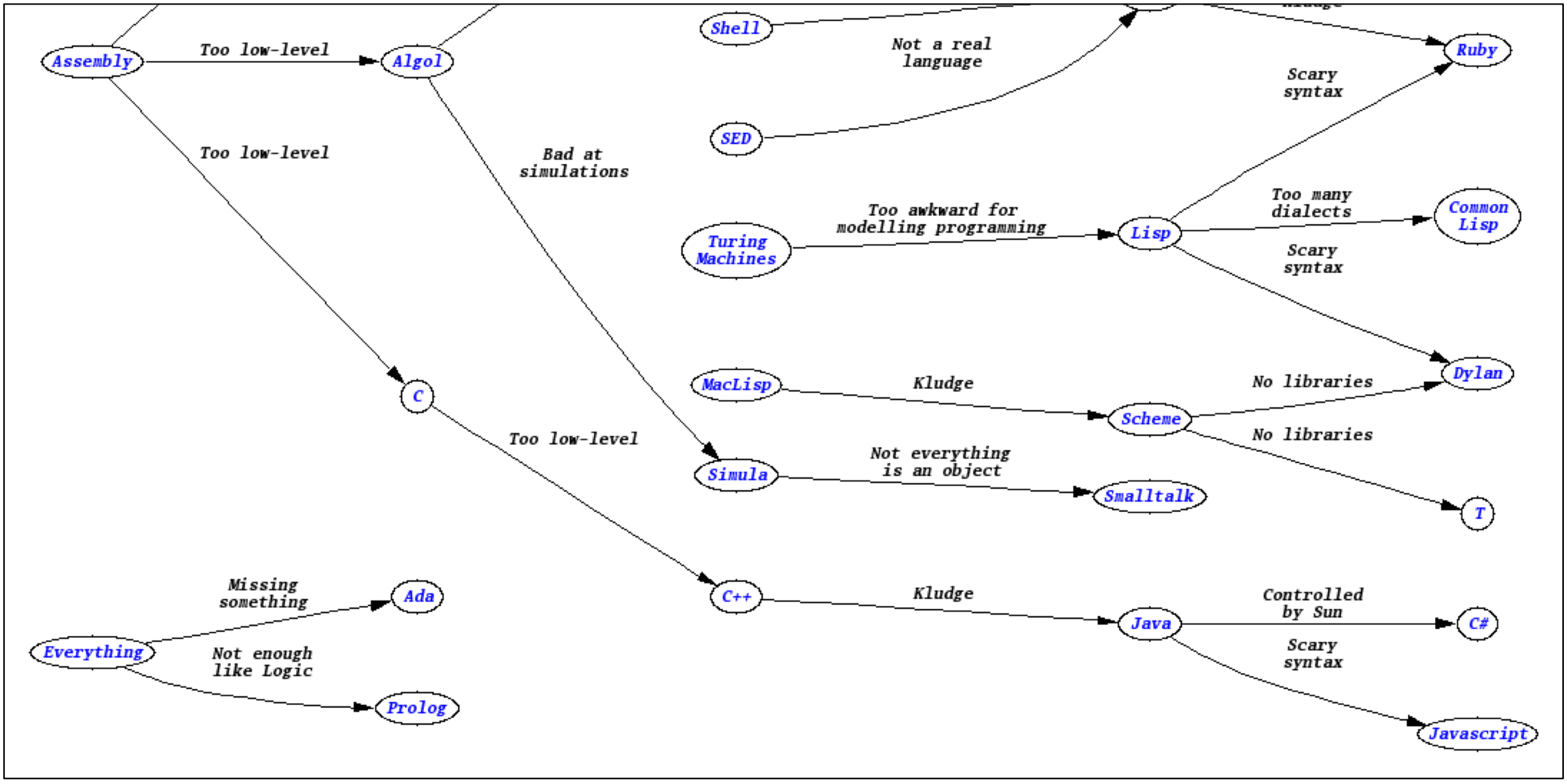
The surprising thing is how many, and how well, languages can be described this way.”

http://www.solipsys.co.uk/new/EveryLanguageFixesSomething.html

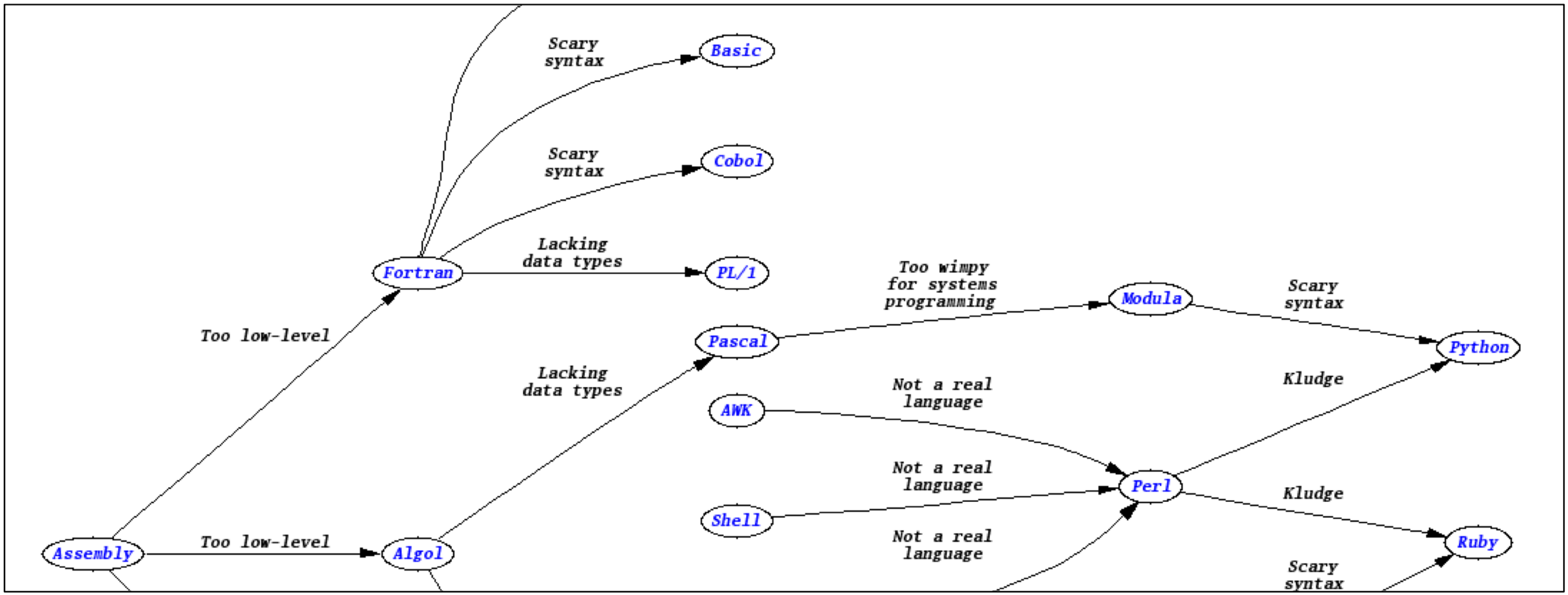
Programming Languages as Reactions



http://www.solipsys.co.uk/new/EveryLanguageFixesSomething.html



http://www.solipsys.co.uk/new/EveryLanguageFixesSomething.html


Scary syntax
Basic
Scary syntax
Cobol
Lacking data types
Fortran
PL/1
Too wimpy for systems programming
Modula
Scary syntax
Python
Too low-level
Pascal
Lacking data types
Not a real language
AWK
Kludge
Not a real language
Perl
Shell
Kludge
Not a real language
Ruby
Too low-level
Assembly
Algol
Scary syntax

Paul Graham's Wish List for a Programming Language

- Lisp was designed in late 1950s.
- It was a radical departure from existing languages e.g. Fortran.
- It embodied nine new ideas, which can be looked upon as a wish list for a programming language.

http://www.paulgraham.com/icad.html

Paul Graham's Wish List for a Programming Language

Wish List	Description / Example
Conditionals	If-the-else constructs are taken for granted now, but Fortran didn’t have them.
A function type	Functions as data type just like integers or strings and can be stored in variables, passed as arguments, etc.
Recursion	The solution to a problem depends on solutions to smaller instances of the same problem i.e. by allowing a function to call itself within the program text.
Dynamic typing	Where values have types, not the variables.
Garbage collection	Automatic memory management by reclaiming memory occupied by objects that are no longer in use.
Programs composed of expressions	As opposed to a series of statements.
A symbol type	Symbols are effectively pointers to strings stored in a hash table. So you can test equality by comparing a pointer, instead of comparing each character.
A notation for code using trees of symbols and constants	e.g. expressing programs directly in parse trees that get built behind the scenes.
The whole language there all the time	i.e. no real distinction between read-time, compile-time and runtime.

http://www.paulgraham.com/icad.html

Java

Wish List
Conditionals
A function type (from Java 8)
Recursion
Dynamic typing
Garbage collection
Programs composed of expressions
A symbol type
A notation for code using trees of symbols and constants
The whole language there all the time

http://www.paulgraham.com/icad.html

Groovy/Ruby/Python/Scala/Xtend/Kotlin

(from Neal Ford)

Wish List
Conditionals
A function type
Recursion
Dynamic typing
Garbage collection
Programs composed of expressions
A symbol type
A notation for code using trees of symbols and constants
The whole language there all the time

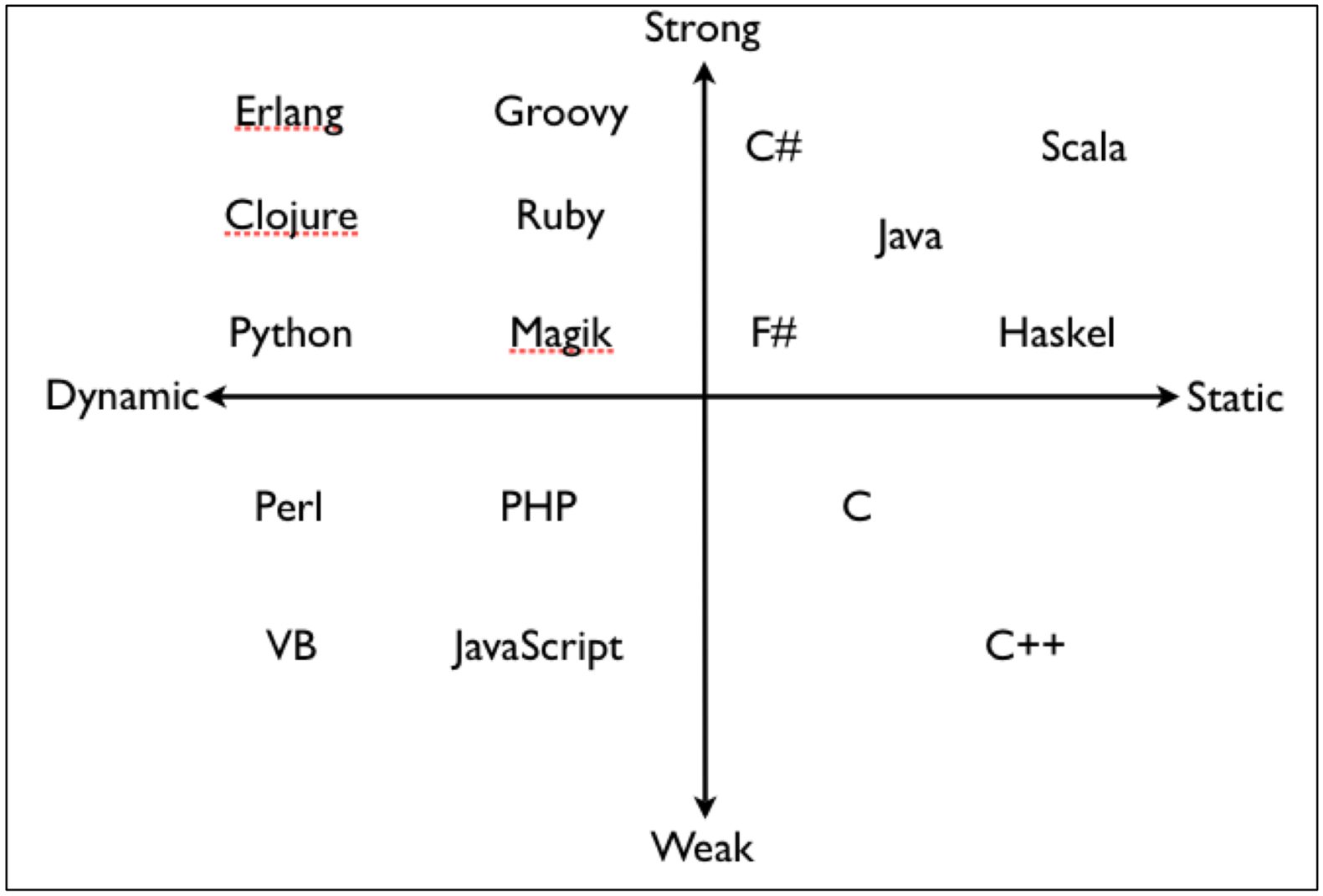
http://www.paulgraham.com/icad.html

Typing

The concept of applying a “type” to a variable

Typing Spectrum

Languages are often classified based on their approach to typing…



Defining Typing Terms...

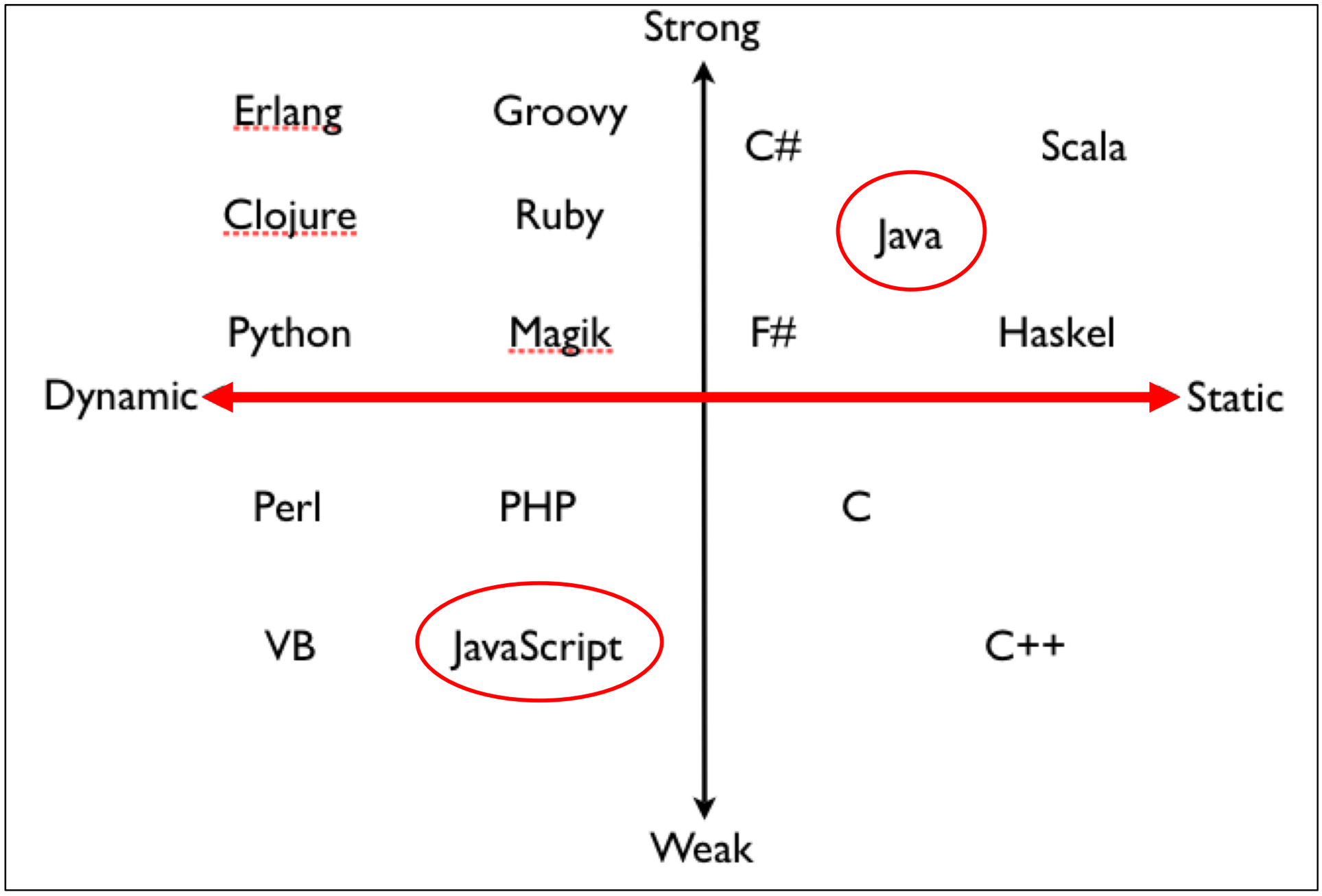
"There is widespread confusion or disagreement about the meanings of the words

static, dynamic, strong and weak

when used to describe the type systems of programming languages"

https://pythonconquerstheuniverse.wordpress.com/2009/10/03/static-vs-dynamic-typing-of-programming-languages/

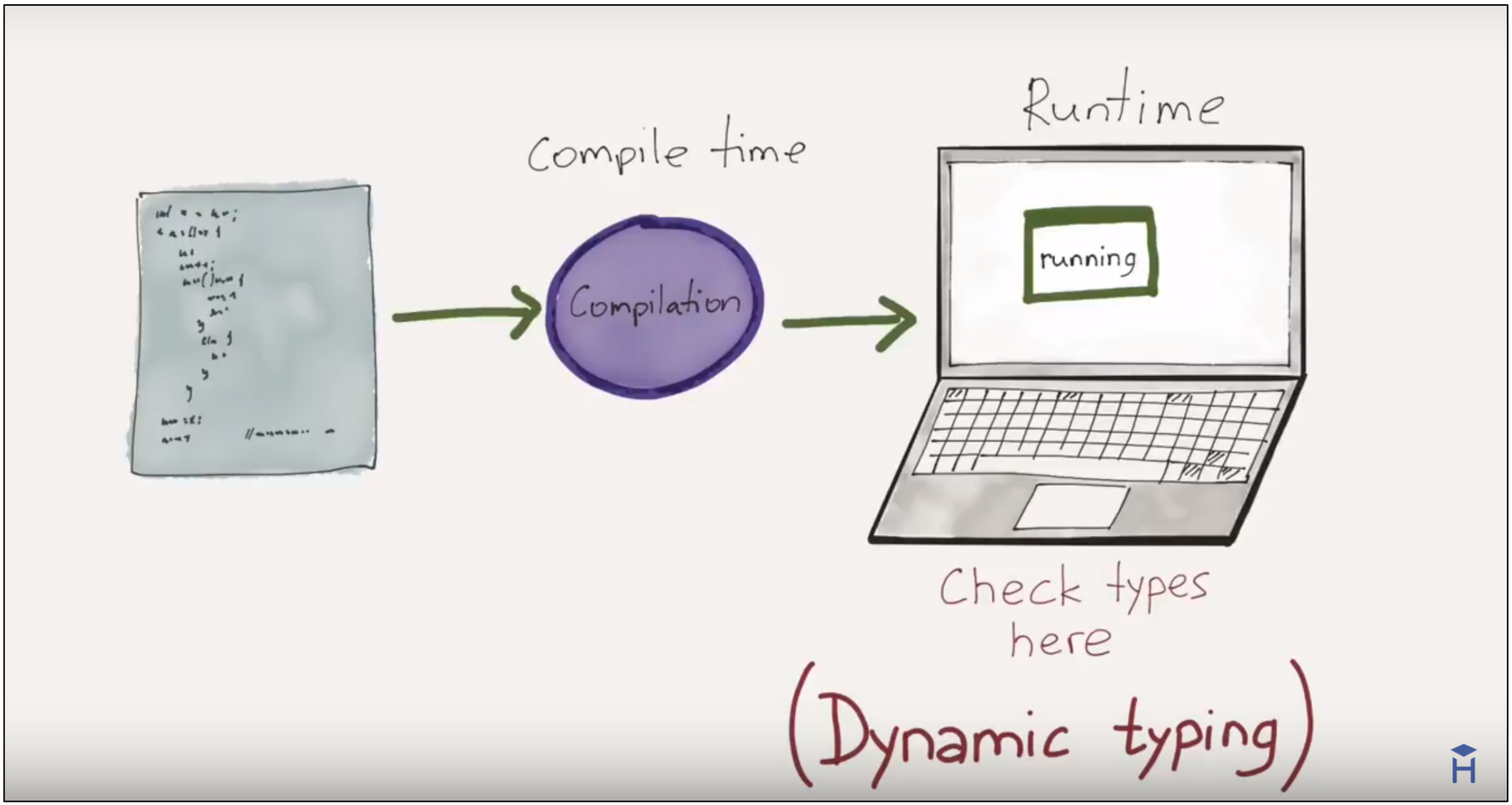

Dynamic
Static


Strong
Erlang
Groovy
C#
Scala
Clojure
Ruby
Java
Python
Magik
F#
Haskel
Dynamic
Static
Perl
PHP
C
VB
JavaScript
C++
Weak

Dynamic Typing

“Variables’ type declarations are not mandatory and they will be generated/inferred on the fly, by their first use.”

http://blogs.agilefaqs.com/2011/07/11/dynamic-typing-is-not-weak-typing/



https://www.youtube.com/watch?v=C5fr0LZLMAs

Dynamic Typing – Example


```
var greeting = ;  //undefined
var someRandomInteger = 100;
var aDoubleVariable = 2.2;
greeting = "Hello!";
```

A type is NOT assigned to the variables.

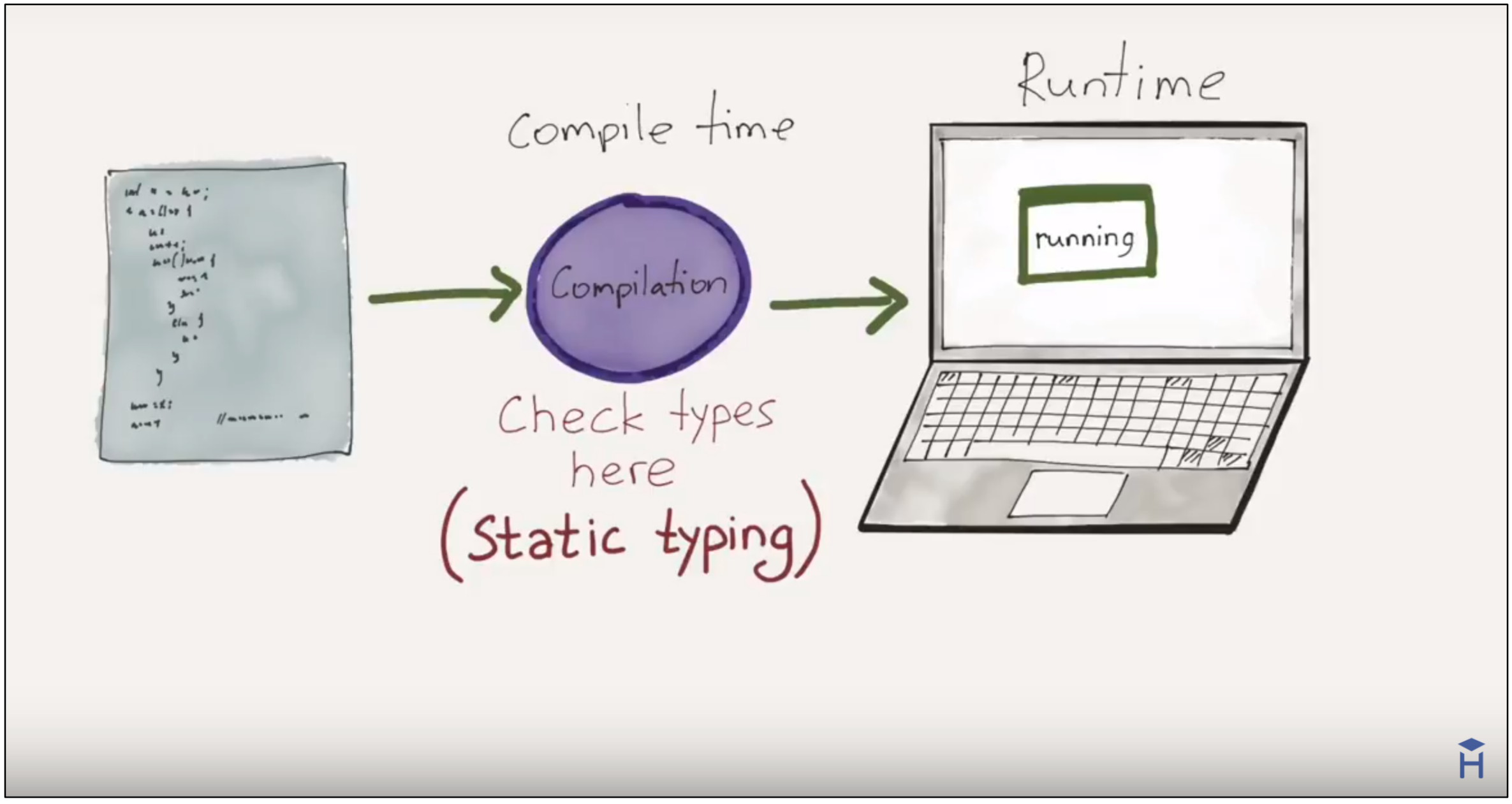
The JavaScript engine will choose a type that it feels best describes the data that's contained inside of your variable → assign datatype behind the scenes.

https://howtoprogramwithjava.com/dynamic-typing-vs-static-typing/

Static Typing

“Variable declarations are mandatory before usage, else results in a compile-time error"

http://blogs.agilefaqs.com/2011/07/11/dynamic-typing-is-not-weak-typing/



https://www.youtube.com/watch?v=C5fr0LZLMAs

Static Typing – Example


```
String greeting = "Hello!";
int someRandomInteger = 100;
double aDoubleVariable = 2.2;
```

A type is assigned to each variable.

In Java, if we don't assign a type, we get a compiler error → Java is statically typed.

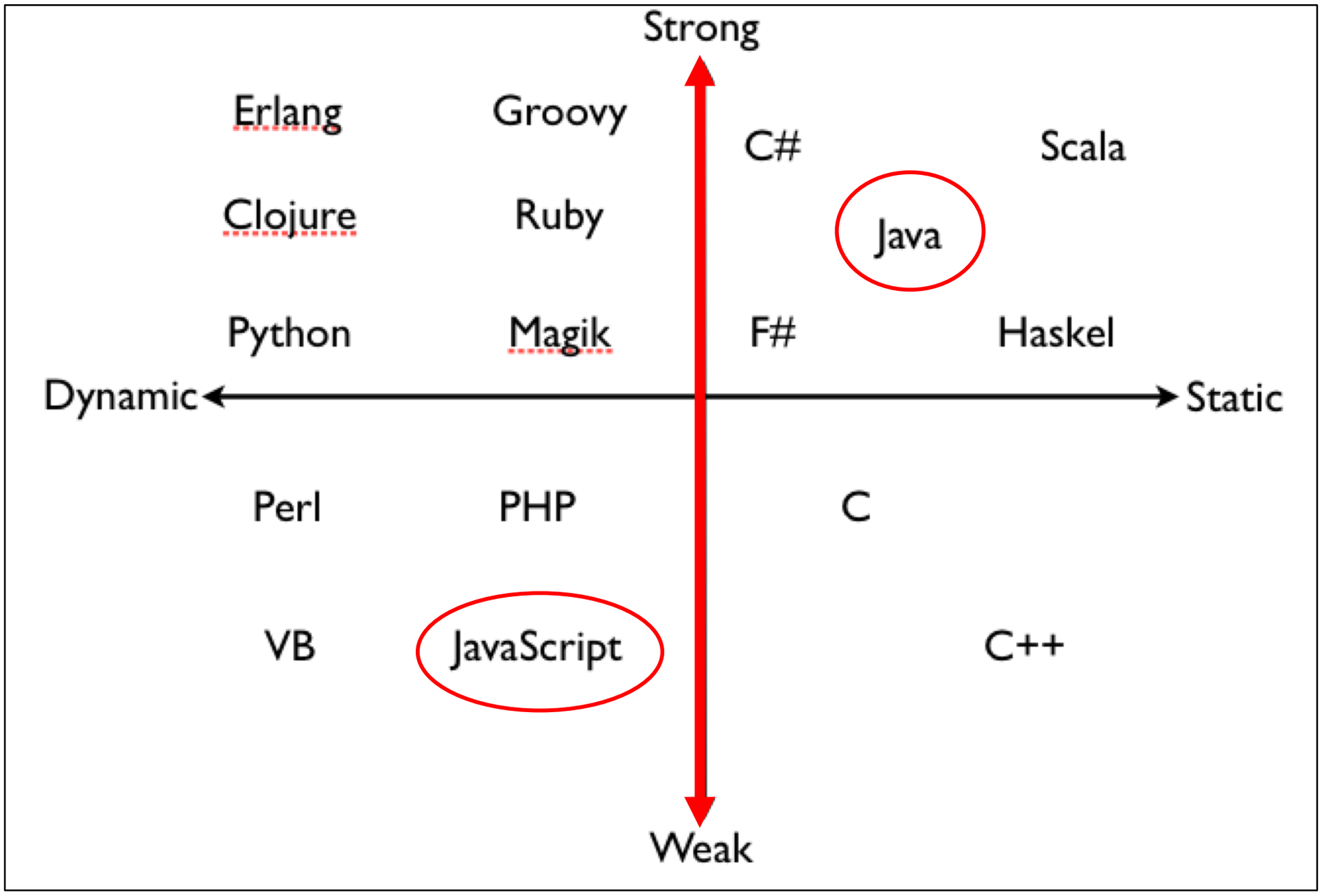
Types determine the operations we can perform on the variables.

https://howtoprogramwithjava.com/dynamic-typing-vs-static-typing/

Amount of type checking enforced by the compiler vs. leaving it to the runtime

http://blogs.agilefaqs.com/2011/07/11/dynamic-typing-is-not-weak-typing/


Strong
Weak


Strong
Erlang
Groovy
C#
Scala
Clojure
Ruby
Java
Python
Magik
F#
Haskel
Dynamic
Static
Perl
PHP
C
VB
JavaScript
C++
Weak

Strong Typing

“Once a variable is declared as a specific data type, it will be bound to that particular data type.

You can explicitly cast the data type though.”

http://blogs.agilefaqs.com/2011/07/11/dynamic-typing-is-not-weak-typing/

Strong Typing – Example 1

*StrongTyping.java

```
1
2 public class StrongTyping {
3
4     public static void main(String[] args) {
5
6         int numberOne = 4;      //static typing
7         int numberTwo;
8
9         numberTwo = 4.6;
10
11    }
12
13 }
14
```

Type mismatch: cannot convert from double to int

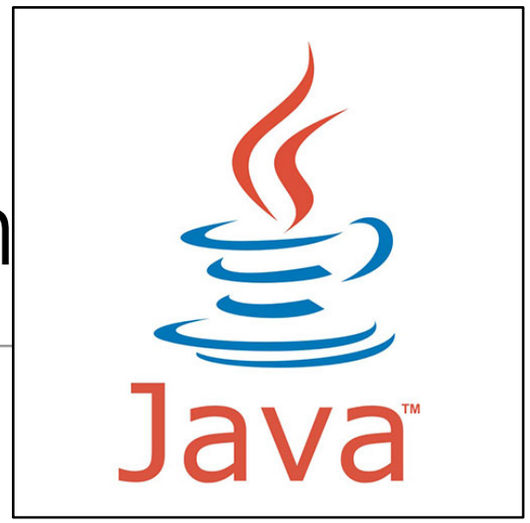
2 quick fixes available:

Add cast to 'int'

Change type of 'numberTwo' to 'double'

Press 'F2' for focus

Strong Typing – Example 1 (fix with castin



*StrongTyping.java

```
1
2  public class StrongTyping {
3
4      public static void main(String[] args) {
5
6          int numberOne = 4;      //static typing
7          int numberTwo;
8
9          numberTwo = (int) 4.6;
10
11     }
12
13 }
```

Casting resolves the type mismatch error

Strong Typing – Example 1 (fix with type)

Changing type to double from int.

*StrongTyping.java

```
1
2  public class StrongTyping {
3
4      public static void main(String[] args) {
5
6          int numberOne = 4;       //static typing
7          double numberTwo;
8
9          numberTwo = 4.6;
10
11     }
12
13 }
```

Strong Typing – Example 2

StrongTyping.java

```
1
2 public class StrongTyping {
3
4     public static void main(String[] args) {
5
6         int a = 4;          //static typing
7         String b = "8";     //static typing
8
9         System.out.print("a * b gives: ");
10        System.out.print(a * b);
11    }
12
13 }
14
```

The operator * is undefined for the argument type(s) int, String

Press 'F2' for focus

Weak Typing

“Variables are not of a specific data type.

However it doesn’t mean that variables are not “bound” to a specific data type.

In weakly typed languages, once a block of memory is associated with an object it can be reinterpreted as a different type of object.”

http://blogs.agilefaqs.com/2011/07/11/dynamic-typing-is-not-weak-typing/

Weak Typing – Example 1

codingground
SIMPLY EASY CODING

Online Javascript Editor (Javascript)

Preview | Embed | index.htm

```
1  <html>
2  <script>
3  function weakTyping() {
4      var a = 4;              //dymanic typing
5      var b = '8';            //dynamic typing
6      document.write("a + b gives:  ");
7      document.write(a + b);  //weak typing
8  }
9  weakTyping();
10 </script>
11 </html>
```

Result

a + b gives: 48

Weak Typing – Example 2

codingground
SIMPLY EASY CODING

Online Javascript Editor (Javascript)

Preview | Embed | index.htm

```
1  <html>
2  <script>
3  function weakTyping() {
4      var a = 4;              //dymanic typing
5      var b = '8';            //dymanic typing
6      document.write("a * b gives:  ");
7      document.write(a * b);  //weak typing
8  }
9  weakTyping();
10 </script>
11 </html>
```

Result

a * b gives: 32

Weak Typing – Example 3

codingground
SIMPLY EASY CODING

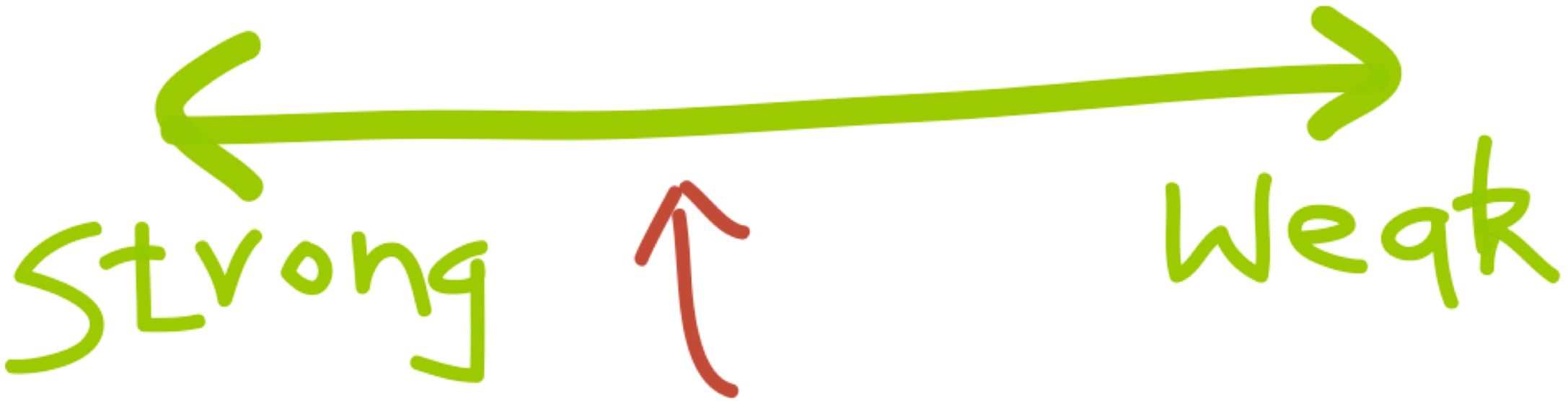
Online Javascript Editor (Javascript)

Preview | Embed | index.htm

```
1  <html>
2  <script>
3  function weakTyping() {
4      var a = 4;                //dymanic typing
5      var b = true;            //dymanic typing
6      document.write("a + b gives:  ");
7      document.write(a + b);  //weak typing
8  }
9  weakTyping();
10 </script>
11 </html>
```

Result

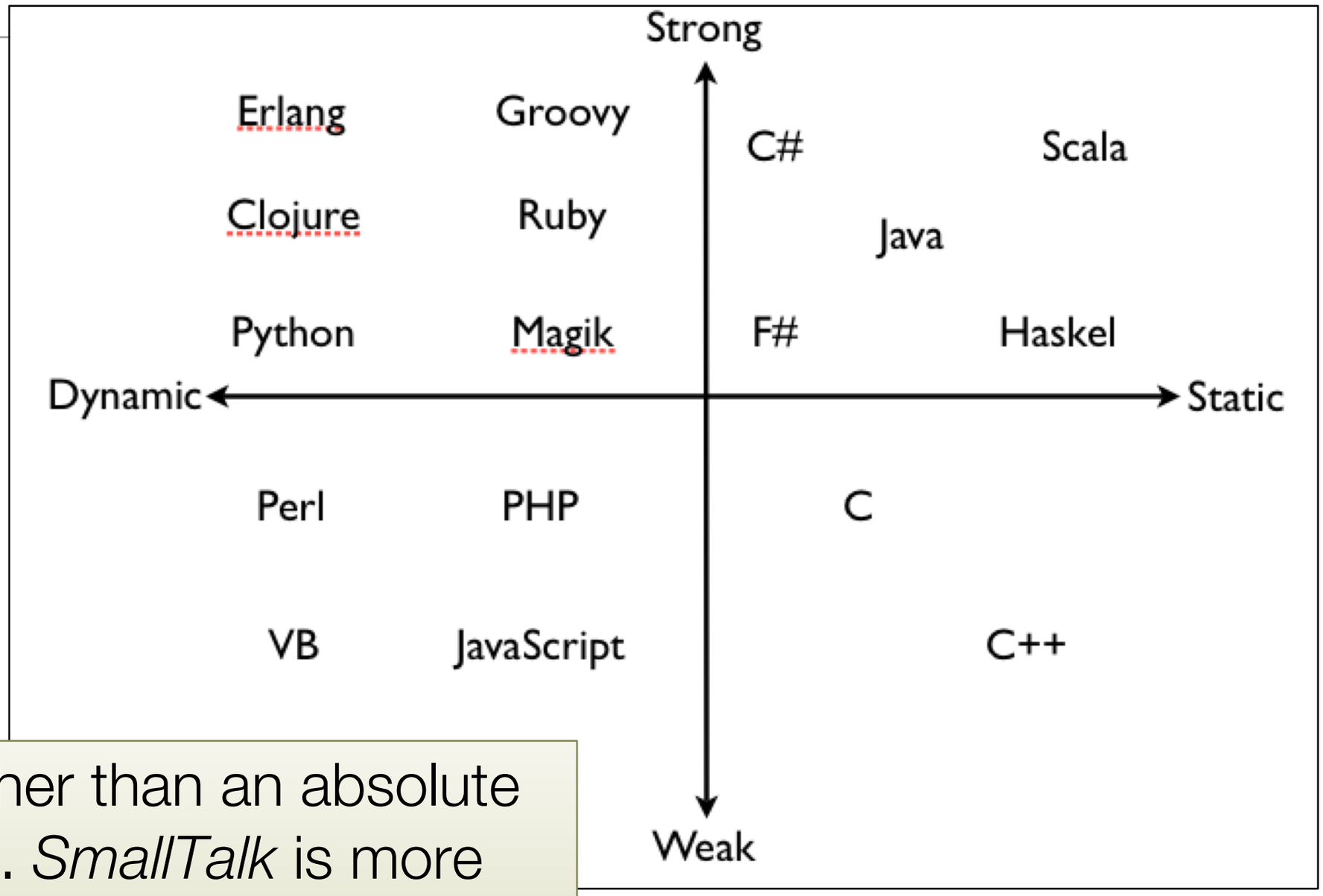
a + b gives: 5



How the runtime constraints you from treating objects of different types (in other words treating memory as blobs or specific data types)

http://blogs.agilefaqs.com/2011/07/11/dynamic-typing-is-not-weak-typing/

Typing Spectrum



Continuum rather than an absolute measure; e.g. *SmallTalk* is more strongly typed compared to *Python* which is more strongly typed than *JavaScript*.

http://blogs.agilefaqs.com/2011/07/11/dynamic-typing-is-not-weak-typing/