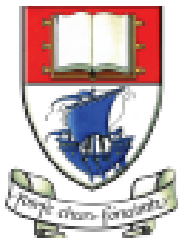


Programming Languages

Produced
by:

Eamonn de Leastar (edeleastar@wit.ie)

Dr. Siobhán Drohan (sdrohan@wit.ie)



Waterford Institute *of* Technology
INSTITIÚID TEICNEOLAÍOCHTA PHORT LÁIRGE

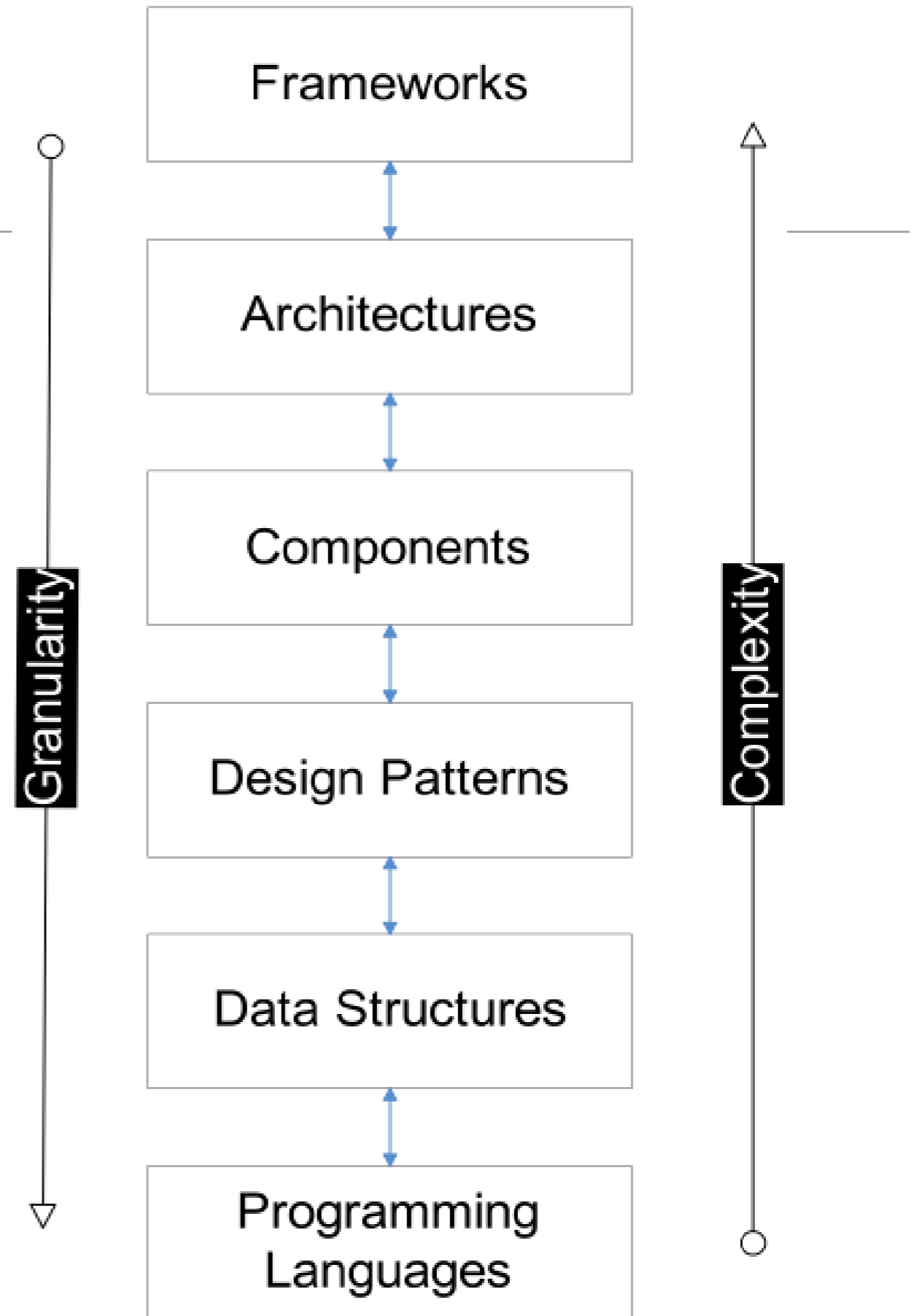
Department of Computing and Mathematics
<http://www.wit.ie/>

Topic List

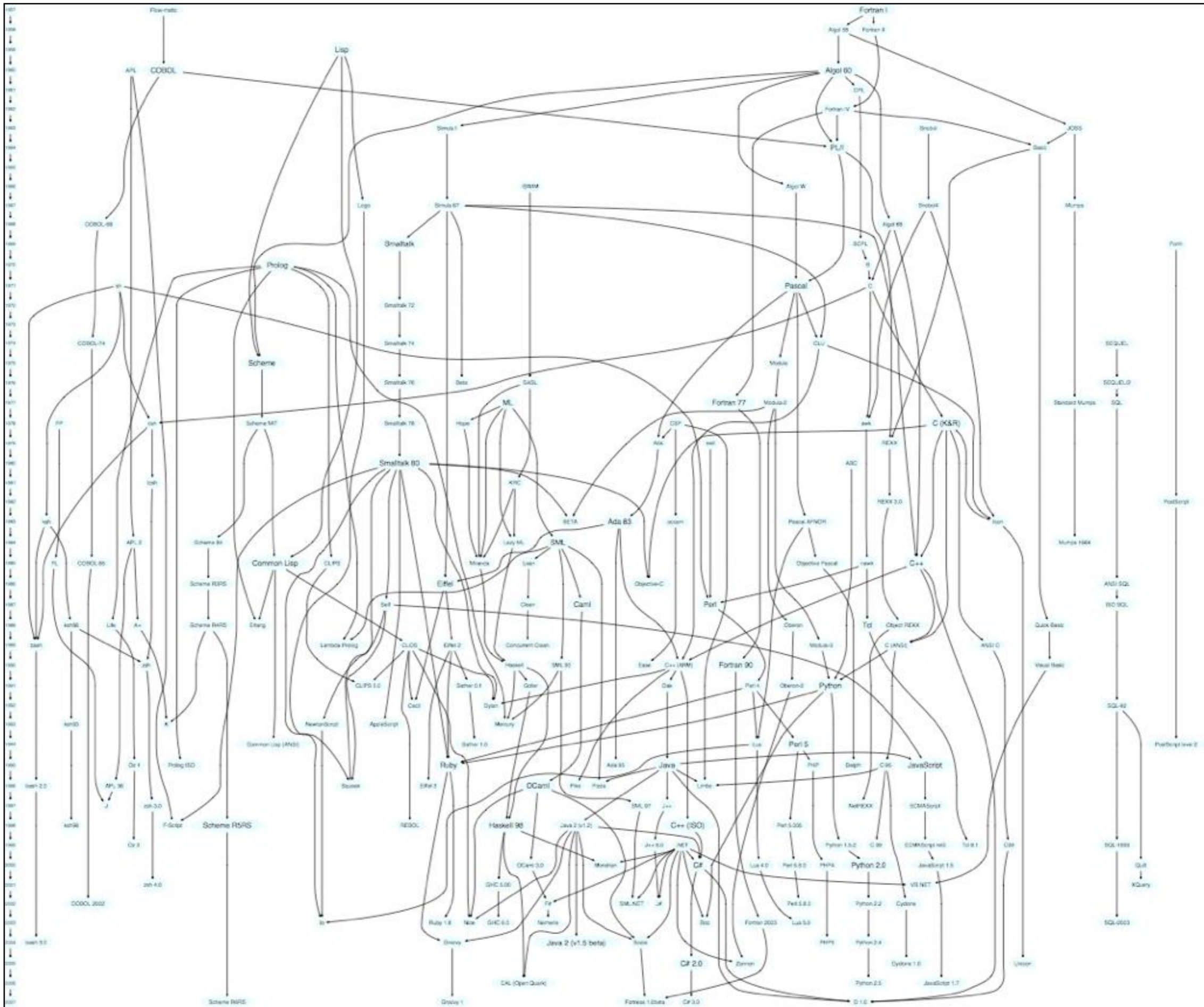
Programming Languages:

- Family Trees.
- Characteristics.
- Typing Spectrum.


Choice of programming language can have profound effect on accessibility of knowledge at higher layers.



Family Tree



Family Tree

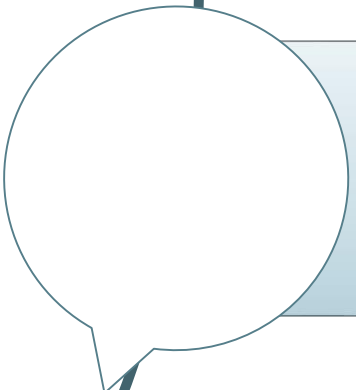


Lisp - designed late 1950s - departure from existing languages (e.g. Fortran).



“Lisp is the Chuck Norris of programming languages.”

Lisp, Fortran Cluster



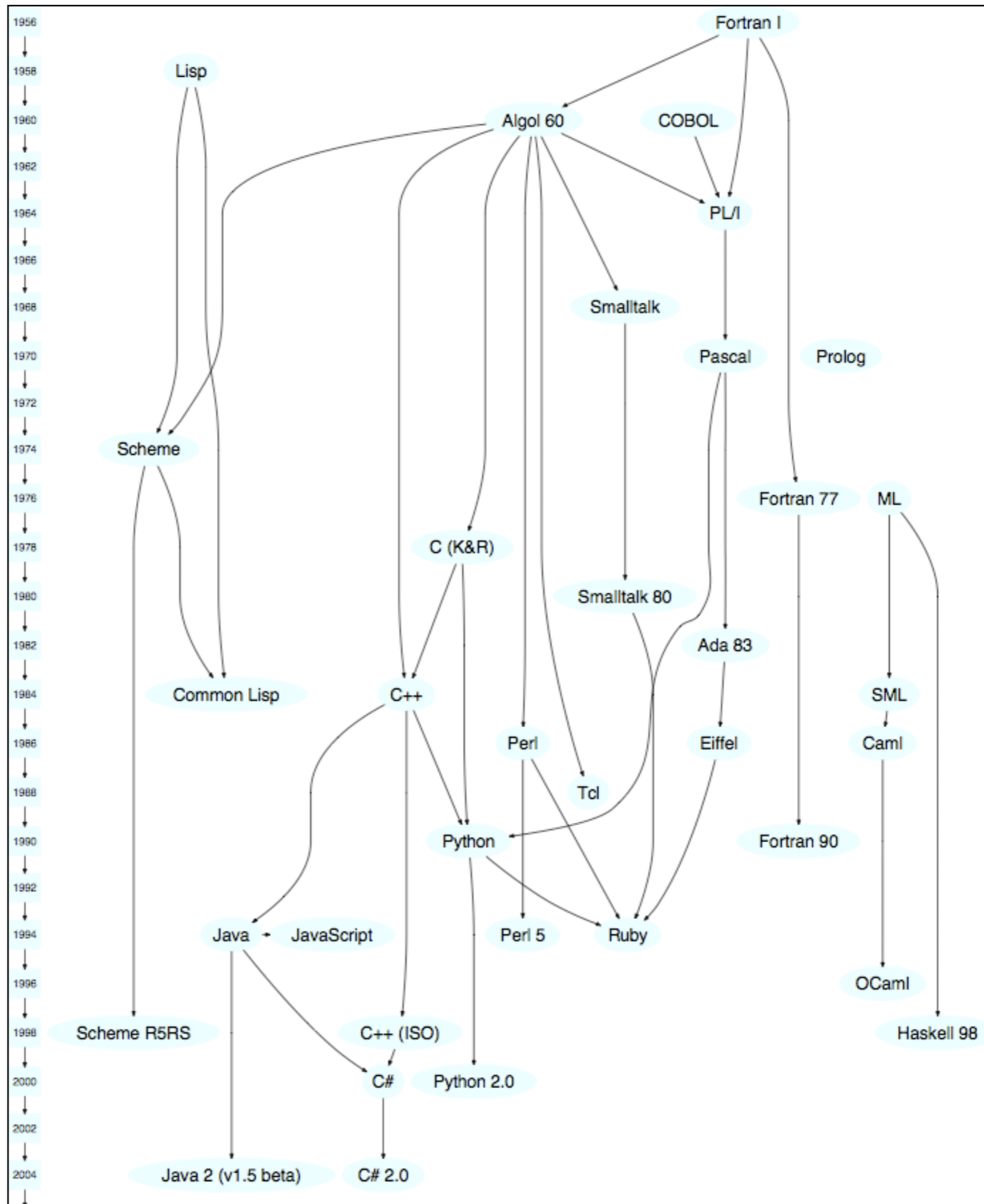
“Lisp is the bar other languages are measured against.”



“Knowing Lisp demonstrates developer enlightenment.”

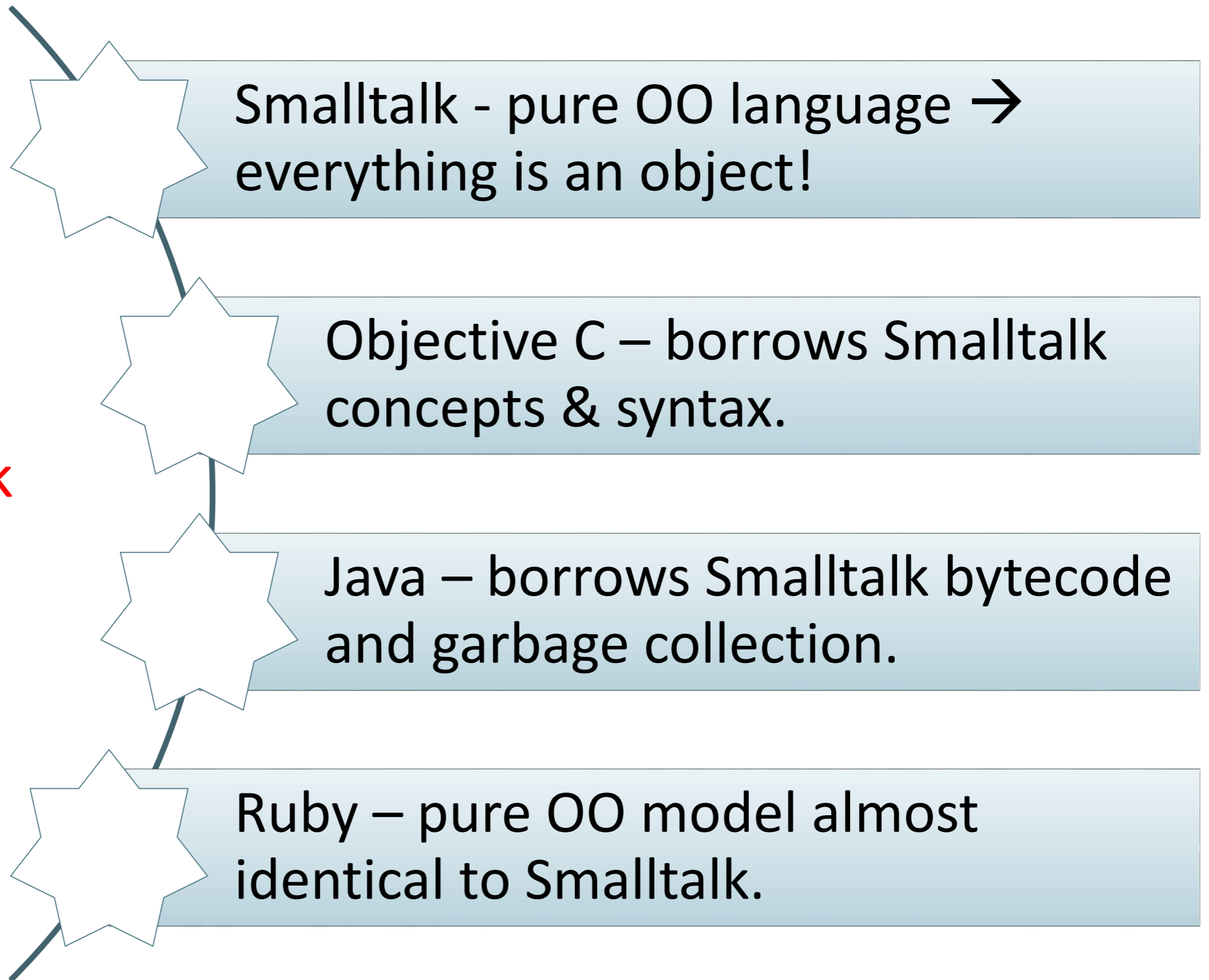
Family Tree

Lisp, Fortran Cluster



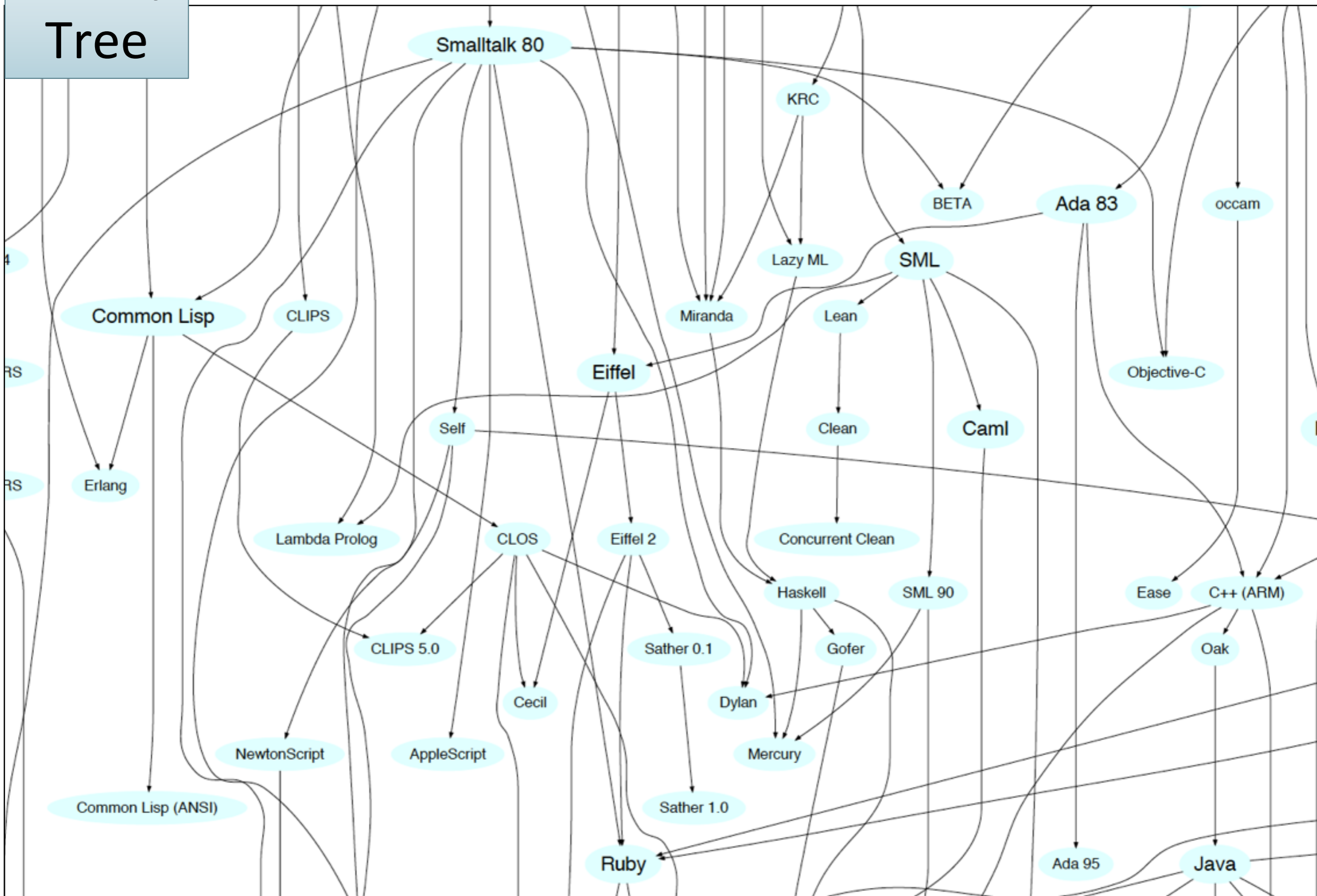
Family Tree

Smalltalk Cluster



Family Tree

Smalltalk Cluster



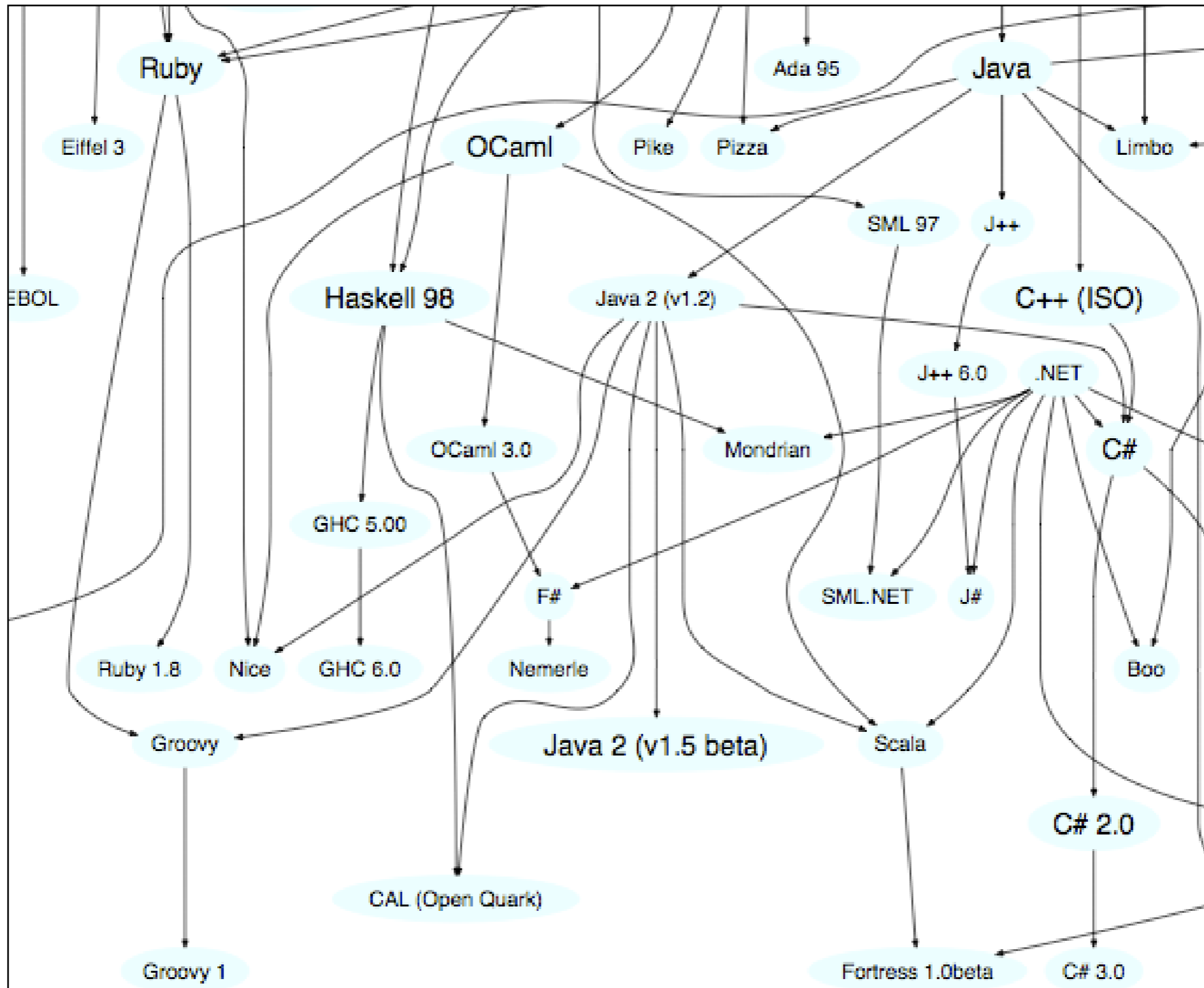
The diagram features a large, light gray arrow pointing to the right. Inside the arrow, there are two rounded rectangular boxes. The left box is dark blue and contains the text 'Smalltalk'. The right box is reddish-brown and contains the text 'Mainstream OO Languages'. The boxes are positioned such that they appear to be part of the arrow's path, suggesting a flow or evolution from the left box to the right box.

Smalltalk

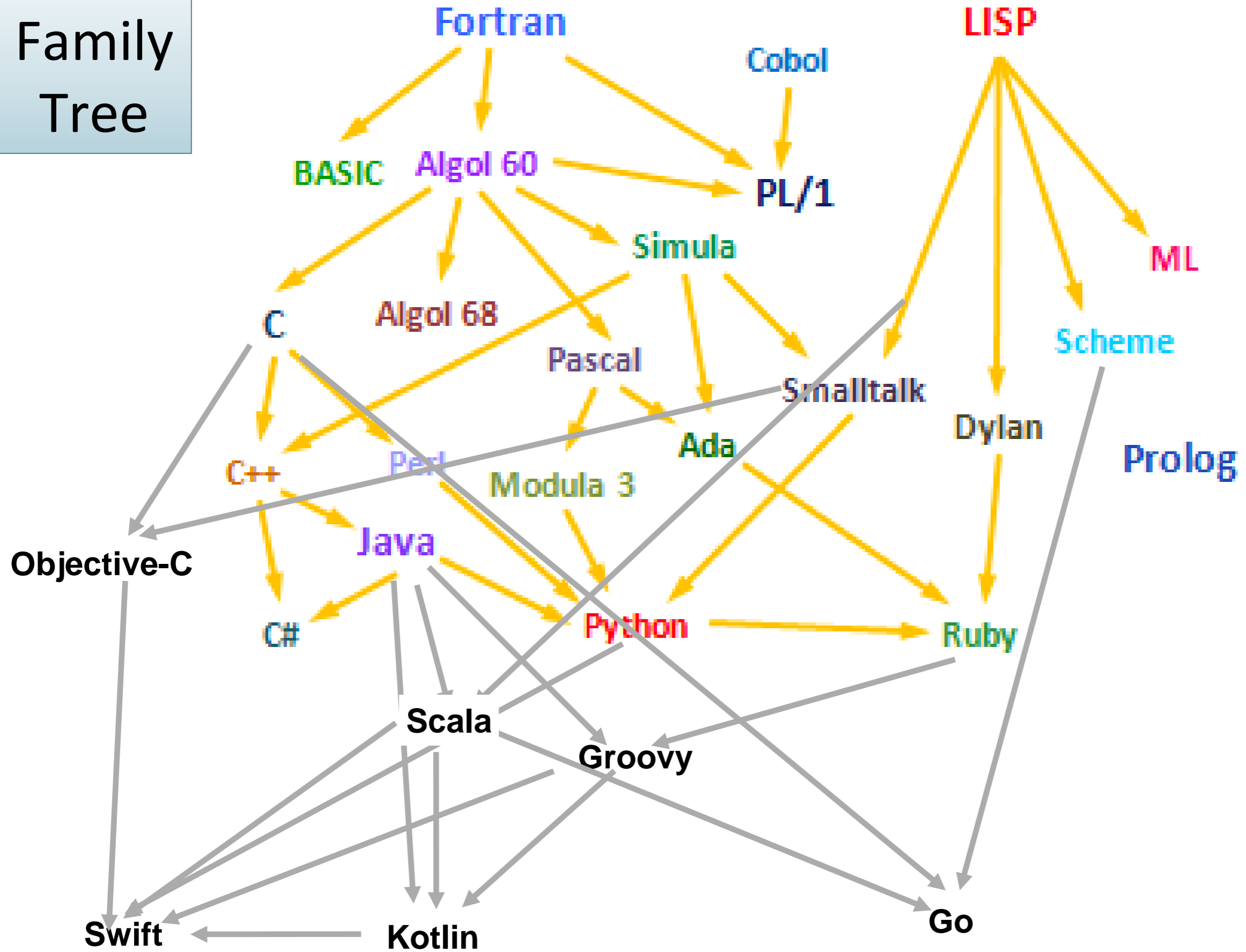
Mainstream
OO Languages

Family Tree

Ruby, Groovy, Java, Scala Cluster



Family Tree



*Why so many
programming
languages?*

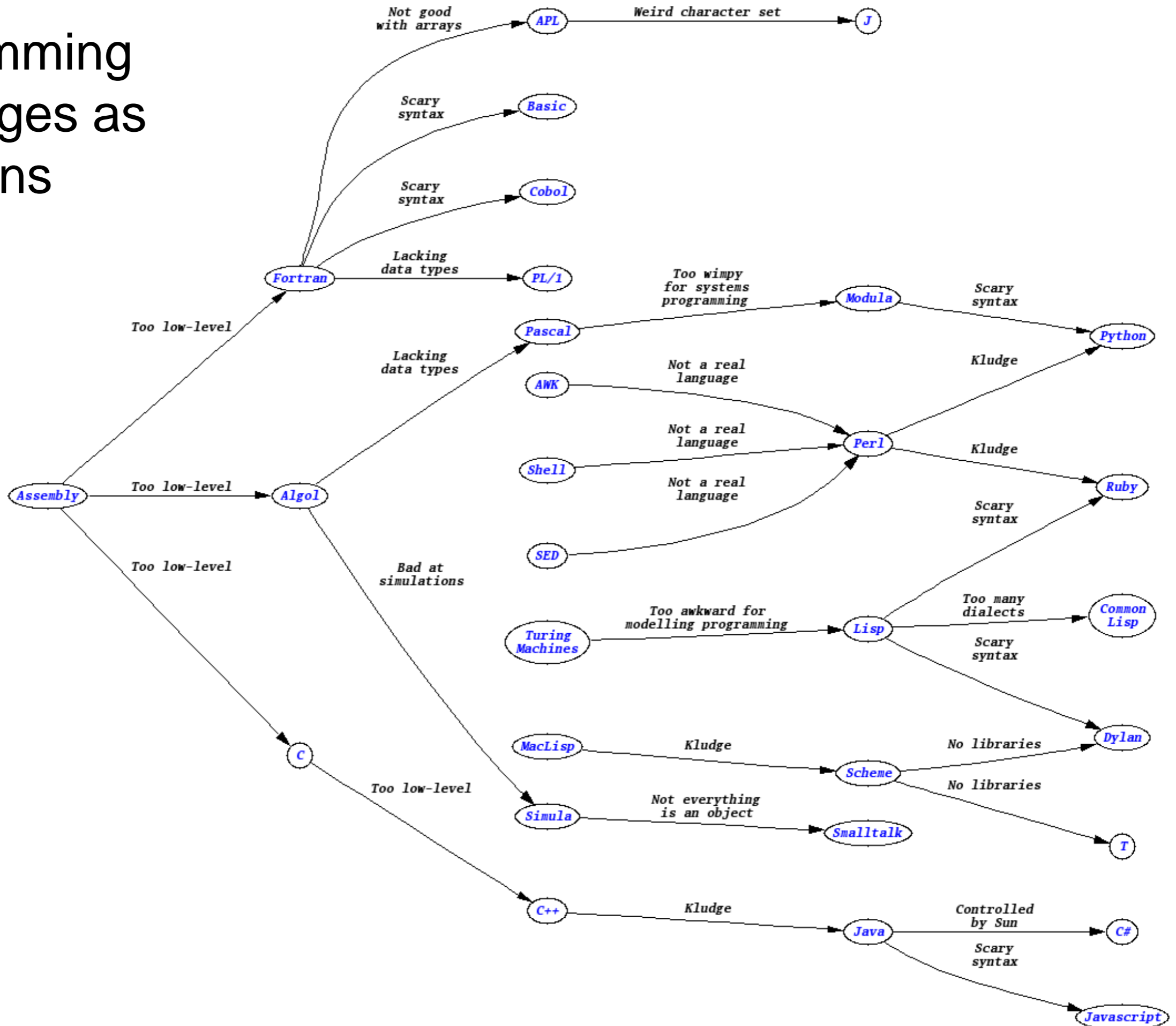
Programming Languages as Reactions

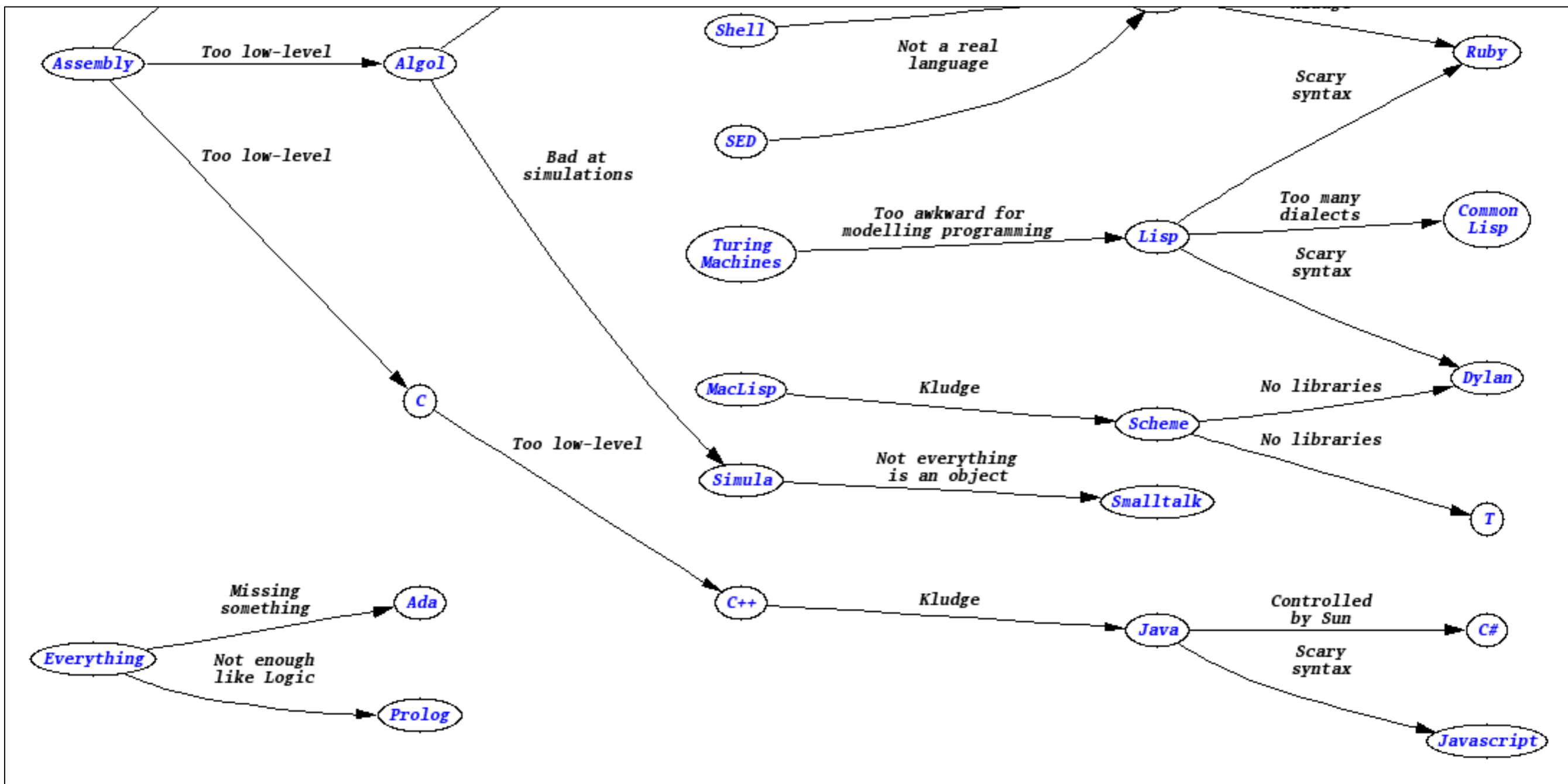
“Kevin Kelleher suggested an interesting way to compare programming languages:

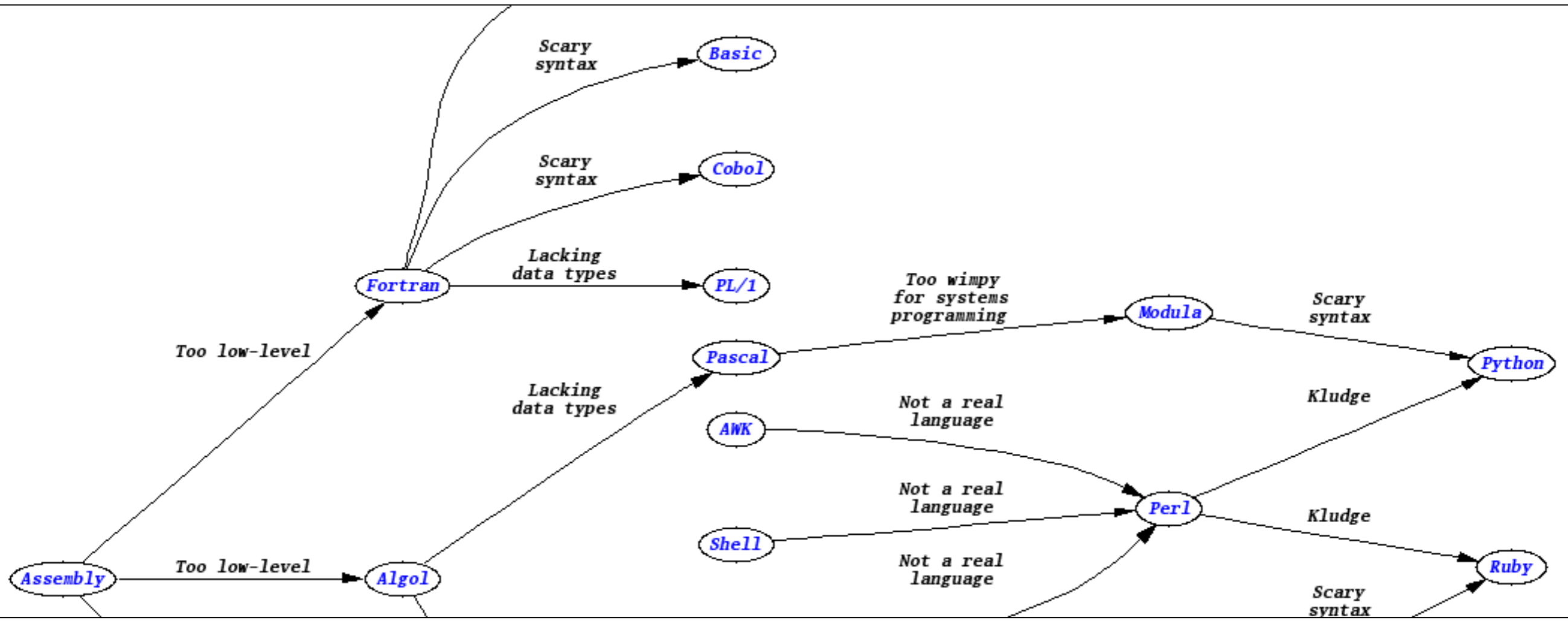
to describe each in terms of the problem it fixes.

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

Programming Languages as Reactions







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.

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.

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

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

+ Metaprogramming

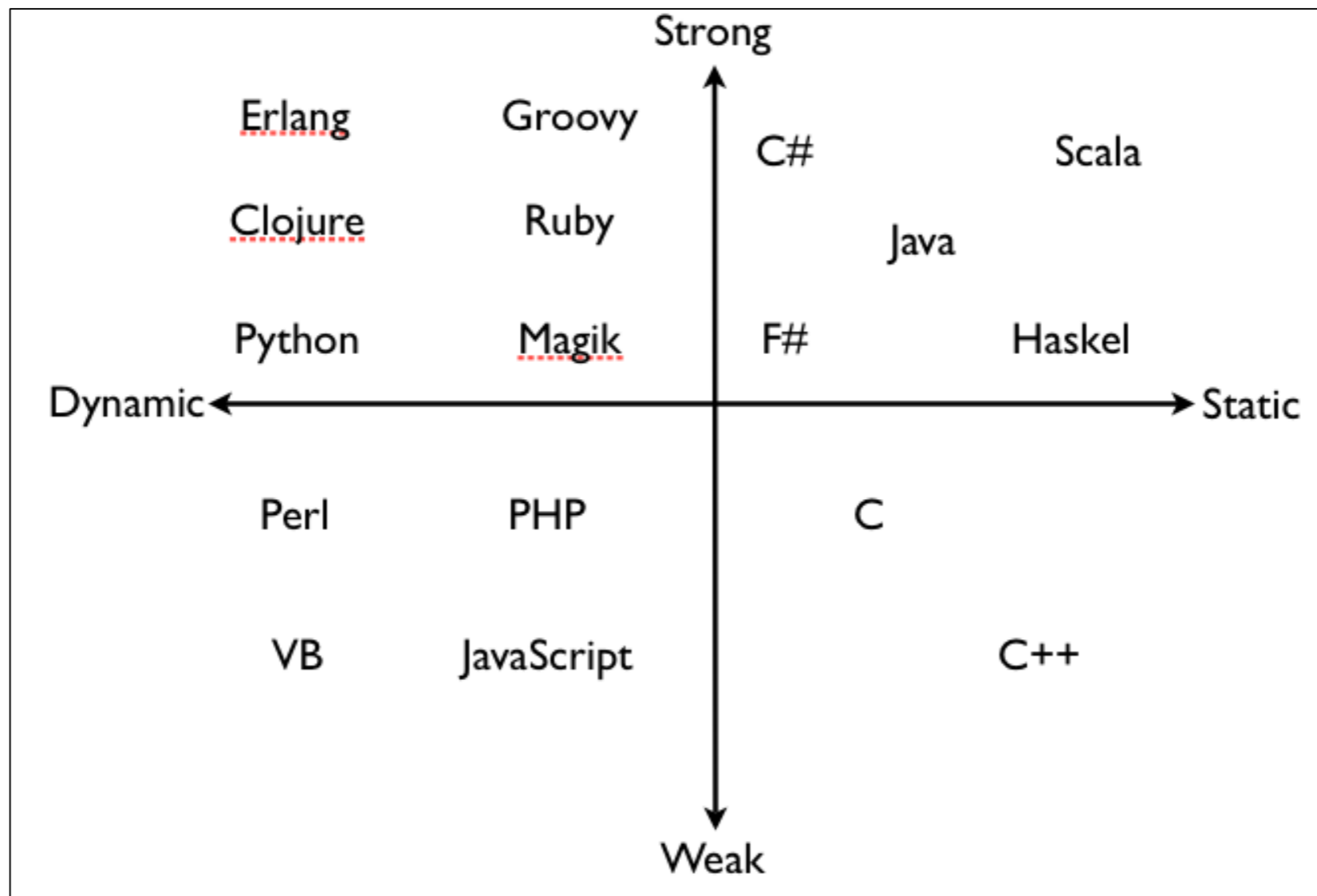
e.g. Xtend can generate Java code on the fly.

Typing

The concept of applying a “type” to a variable

Typing Spectrum

Languages are often classified based on their approach to typing...



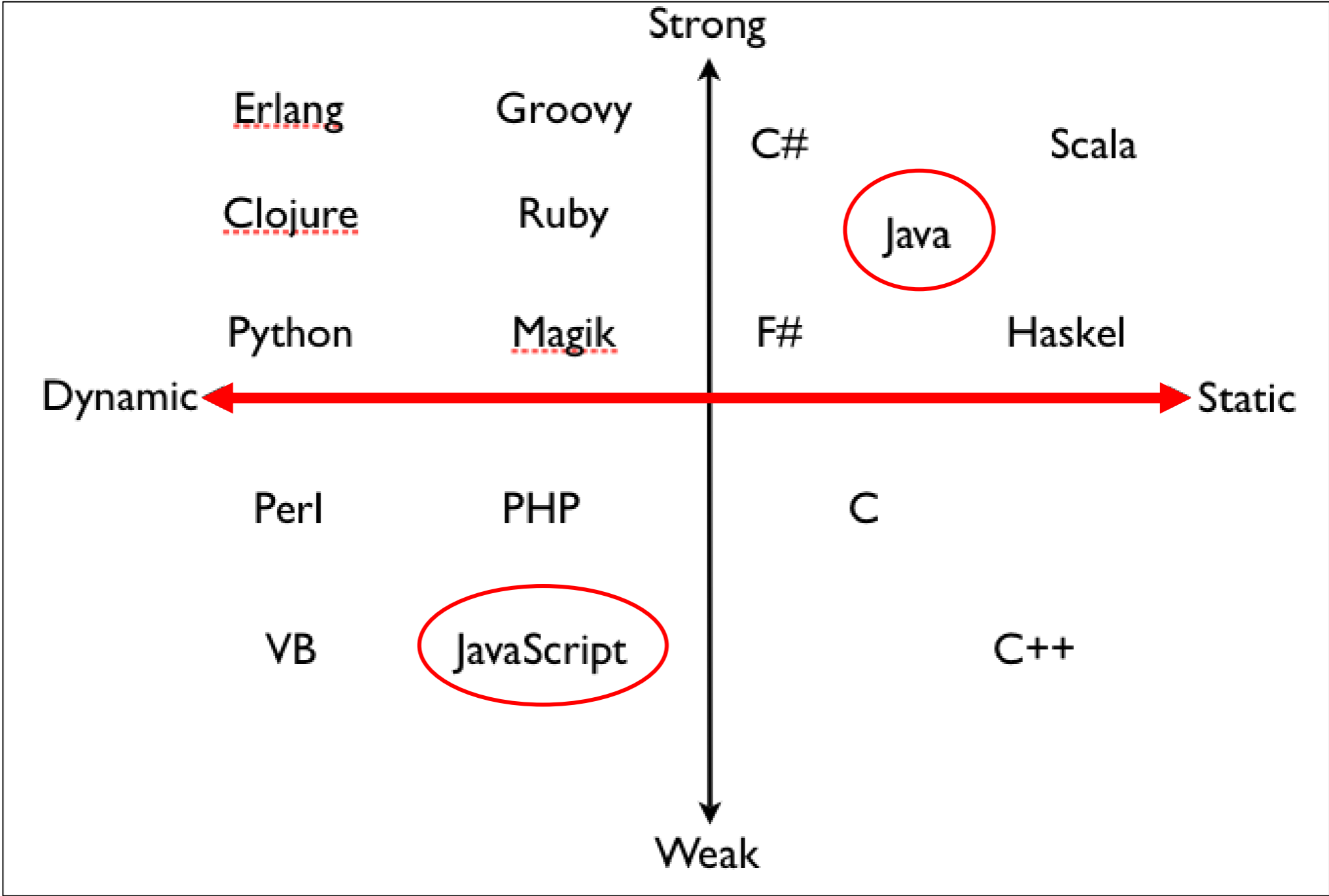
Defining Typing Terms...

“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”





Dynamic Typing

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

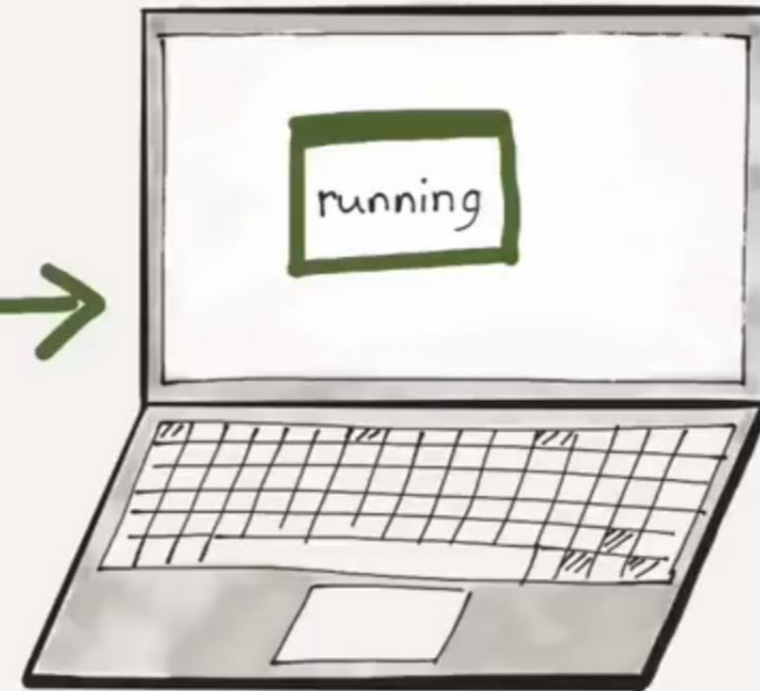
```
int a = 40;
call()
{
  int
  sum;
  sum()sum()
  sum
  }
}
//main.cpp
```



Compile time



Runtime



Check types here

(Dynamic typing)



Dynamic Typing – Example



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

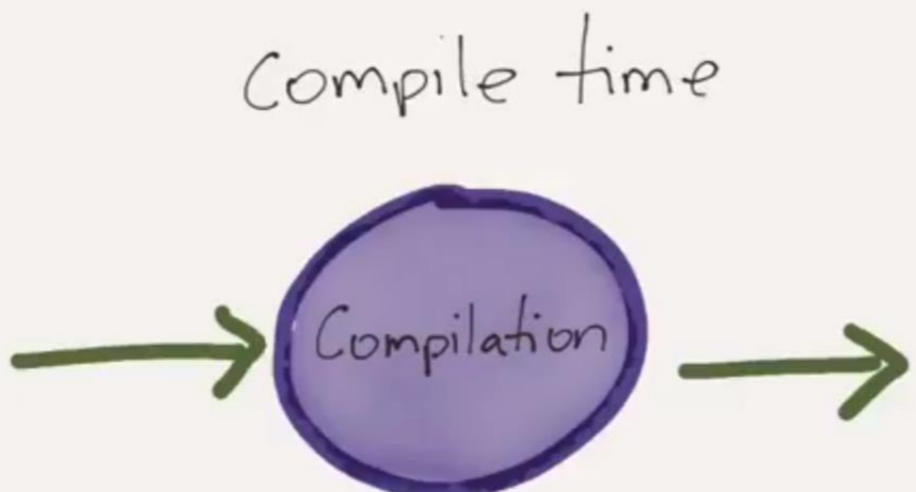
A type is NOT assigned to the variables.

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.

Static Typing

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

```
int a = 40;  
char *str = "hello";  
let  
main:  
    int i;  
    for(i=0; i<10; i++)  
        printf("%d\n", i);  
    }  
}
```

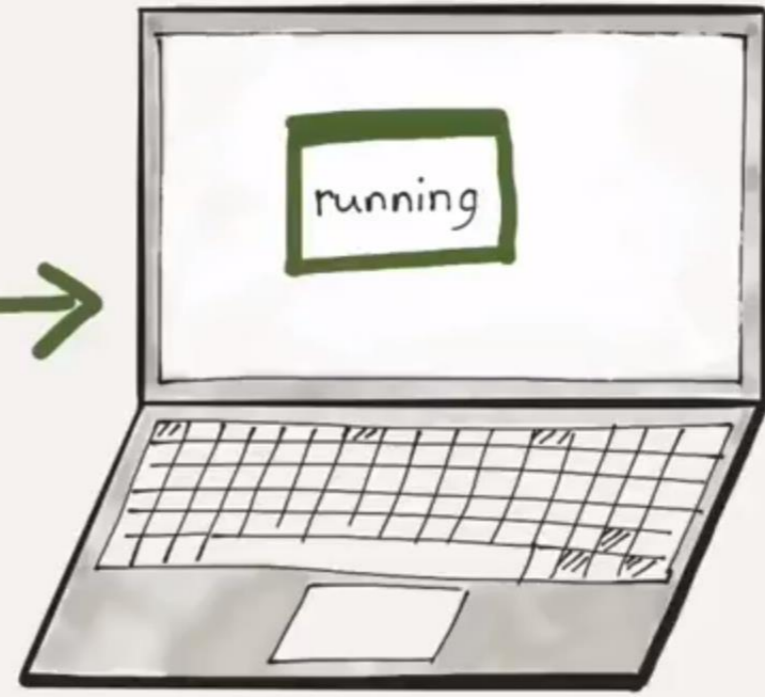


Compile time

Compilation

Check types here
(Static typing)

Runtime



Static Typing – Example



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

A type is
assigned to
each variable.

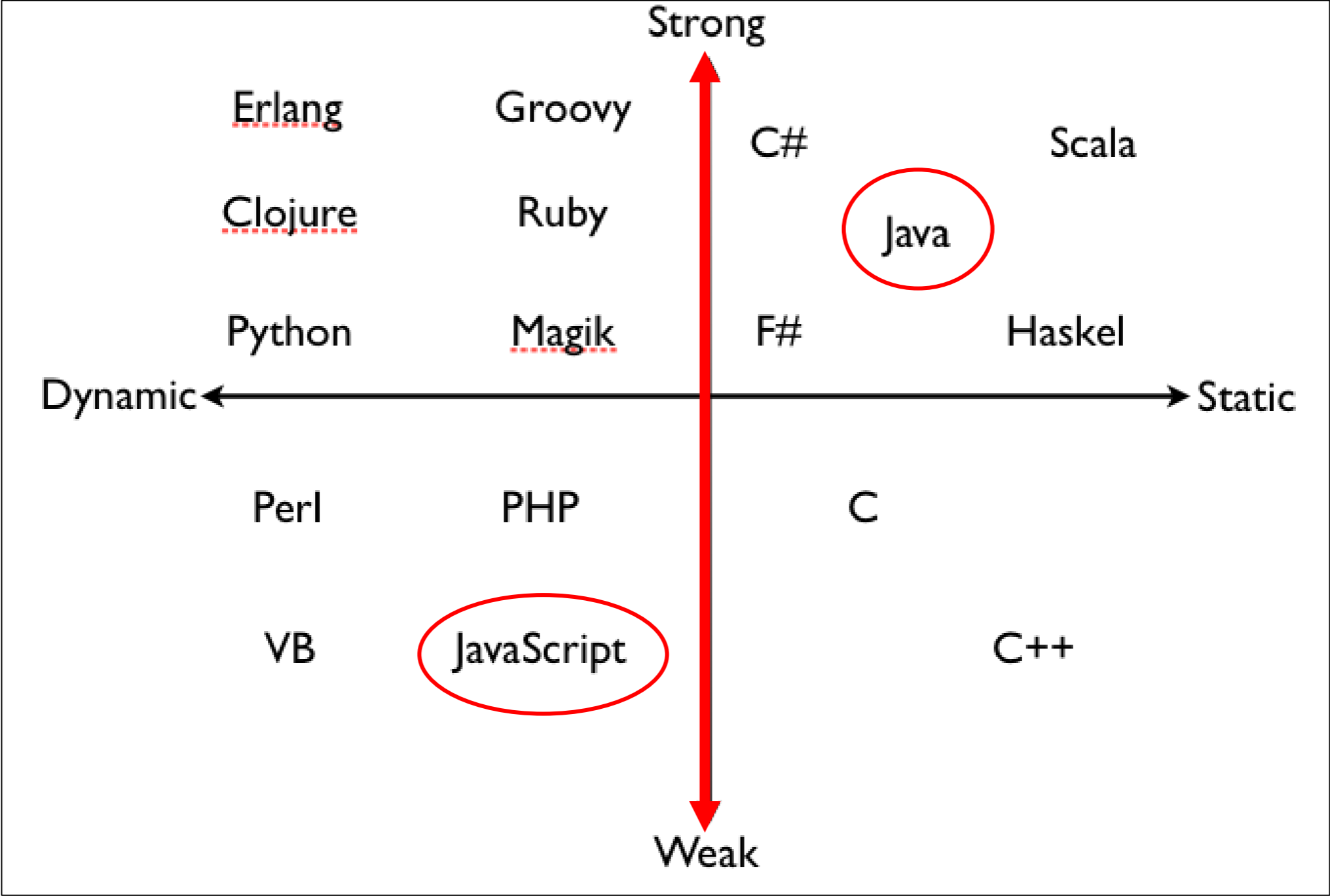
In Java, if we don't assign a
type, we get a compiler error
→ Java is statically typed.

Types determine the
operations we can perform on
the variables.



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





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.”

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 casting)



```
*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 x
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.”

Weak Typing – Example 1



 **codingground** | Online Javascript Editor (Javascript) 

Preview | Embed | index.htm



```
1 <html>
2 <script>
3 function weakTyping() {
4     var a = 4;           //dynamic 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** | Online Javascript Editor (Javascript) 

Preview | Embed | index.htm



```
1 <html>
2 <script>
3 function weakTyping() {
4     var a = 4;           //dynamic 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: 32

Weak Typing – Example 3



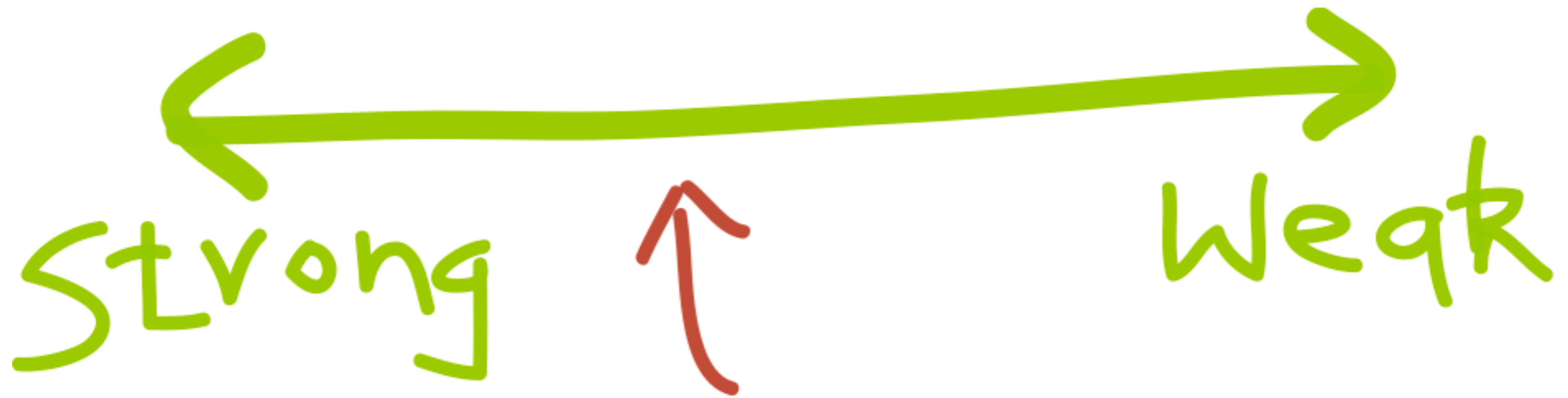
 **codingground** | Online Javascript Editor (Javascript) 

Preview | Embed | index.htm

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

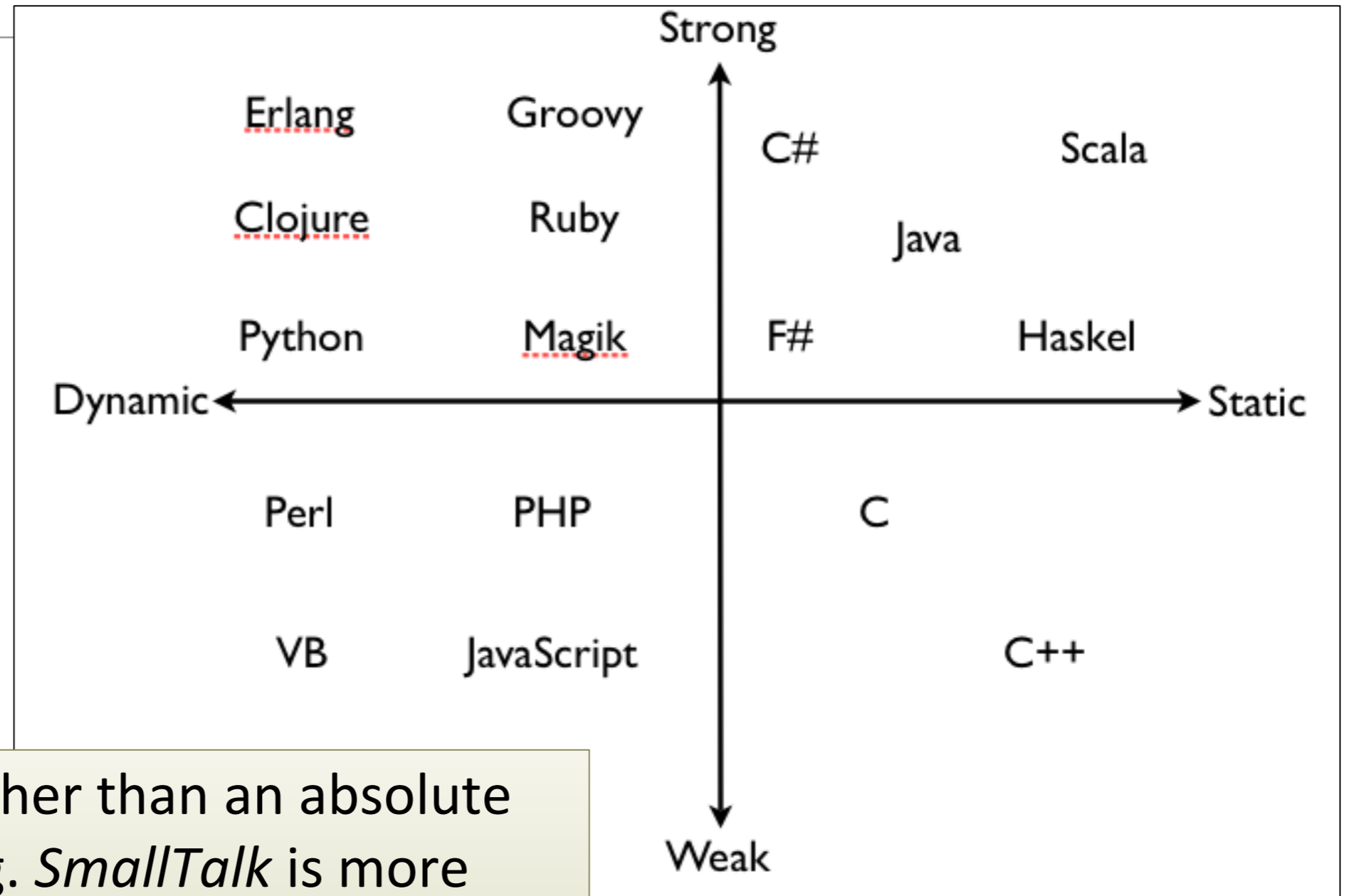
Result

a + b gives: 5

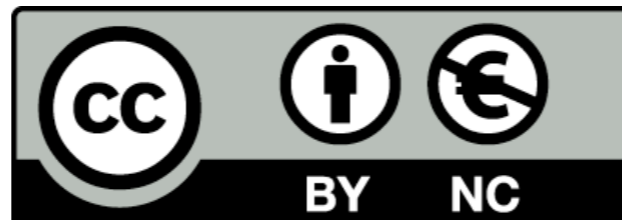


How the runtime constraints you from treating objects of different types (in other words treating memory as blobs or specific data types)

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*.



Except where otherwise noted, this content is licensed under a [Creative Commons Attribution-NonCommercial 3.0 License](http://creativecommons.org/licenses/by-nc/3.0/).

For more information, please see <http://creativecommons.org/licenses/by-nc/3.0/>

