Linguagens e Ambientes de Programação

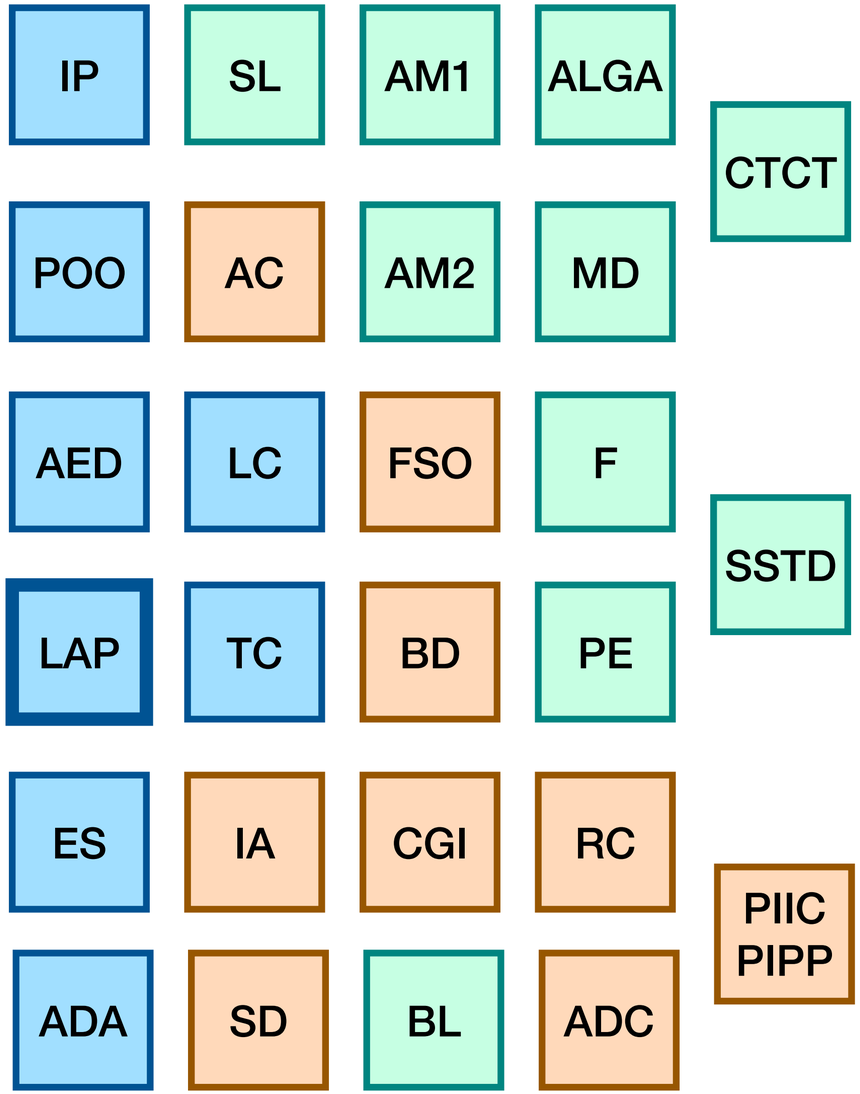
LEI - Licenciatura em Engenharia Informática

João Costa Seco (joao.seco@fct.unl.pt)

NOVA SCHOOL OF
SCIENCE & TECHNOLOGY

LAP 2024
A Unidade Curricular

LAP na LEI


IP
SL
AM1
ALGA
CTCT
POO
AC
AM2
MD
AED
LC
FSO
F
SSTD
LAP
TC
BD
PE
ES
IA
CGI
RC
PIIC
PIPP
ADA
SD
BL
ADC

Linguagens e Ambientes de Programação 2024

1. Estudo de um novo paradigma: programação funcional
2. Estudo dos fundamentos das linguagens de programação
3. A programação funcional é o veículo de comunicação para as LP
4. Aprender Programação Funcional com OCaml
5. Aprender muitas linguagens de programação numa só:
   - fazendo, e comparando.

Porquê estudar linguagens de programação?

Why (1): As linguagens de programação são a “nossa ferramenta”.

Why (2): Cada linguagem é apropriada a um estilo de programas e sistemas. É preciso conhecer bem, para escolher melhor.

Why (3): As linguagens de programação modernas usam/misturam muitos conceitos diferentes (paradigmas).

Why (4): As linguagens também são modelos precisos de raciocínio (programas que simulam sistemas provam que eles são possíveis).

Why (5): Desenhar novas linguagens é mais comum do que se pensa.

Why (6): Quem sabe o porquê das coisas, faz melhor.

Porquê estudar programação funcional (tipificada)?

Why (1): as LP funcionais têm uma semântica precisa

Why (2): os sistemas de tipos permitem raciocínios avançados

Why (3): são linguagens sem interferências e casos especiais

(e.g. visibilidade de nomes, aliasing, efeitos laterais, conversões implícitas).

Why (4): ideais para compreender e desenhar outras linguagens

Why (5): imutabilidade de estado (é possível raciocinar sem ser por debug).

Why (6): modularidade e composicionalidade

Why (7): abstração de ordem superior (código genérico).

Why (8): segurança de tipos (não há nulls, não há (tantos) erros de execução).

Conceitos de linguagens

- Sintaxe, semântica e pragmática
- Linguagens declarativas vs linguagens imperativas
- Declaração e definição de nomes (ligação, ambiente, âmbito)
- Polimorfismo paramétrico e universal
- Abstração e parametrização / Ligação (closures)
- Modularidade e composição
- Tipos algébricos (e indutivos)
- Sistemas de tipos (soundness), inferência de tipos (linguagens com …)
- Raciocínio e correção (testes vs verificação)
- Modelo de execução (stack based) - tail recursive
- Lazy vs Strict
- Estruturas de dados persistentes (why?)
- Estado, aliasing, etc. (pitfalls)
- Interpretadores e Compiladores (Just-in-time)

Programação Funcional em Java

```java
Collections.sort(numbers, new Comparator<Integer>() {
    @Override
    public int compare(Integer n1, Integer n2) {
        return n1.compareTo(n2);
    }
});
```

```java
Collections.sort(numbers, (n1, n2) -> n1.compareTo(n2));
```

https://www.baeldung.com/java-functional-programming

Programação Funcional em C

```c
#include <stdio.h>
#include <stdlib.h>

int values[] = { 88, 56, 100, 2, 25 };

int cmpfunc (const void * a, const void * b) {
   return ( *(int*)a - *(int*)b );
}

int main () {
   int n;

   printf("Before sorting the list is: \n");
   for( n = 0 ; n < 5; n++ ) {
      printf("%d ", values[n]);
   }

   qsort(values, 5, sizeof(int), cmpfunc);

   printf("\nAfter sorting the list is: \n");
   for( n = 0 ; n < 5; n++ ) {
      printf("%d ", values[n]);
   }

   return(0);
}
```

https://www.tutorialspoint.com/c_standard_library/c_function_qsort.htm

Programação Funcional em Scala

```scala
case class ChessPiece(color: Color, name: Name)
val rook = ChessPiece(White, Rook)
```

```scala
sealed trait Color
final case object White extends Color
final case object Black extends Color

sealed trait Name
final case object Pawn extends Name
final case object Rook extends Name
final case object Knight extends Name
final case object Bishop extends Name
final case object Queen extends Name
final case object King extends Name
```

```scala
def isTheMostImportantPiece(c: ChessPiece): Boolean = c match {
    case ChessPiece(_, King) => true
    case _ => false
}
```

https://www.baeldung.com/scala/algebraic-data-types

Programação Funcional em Java >= 17

```java
Object o = ...; // any object
String formatted = null;
if (o instanceof Integer i) {
    formatted = String.format("int %d", i);
} else if (o instanceof Long l) {
    formatted = String.format("long %d", l);
} else if (o instanceof Double d) {
    formatted = String.format("double %f", d);
} else {
    formatted = String.format("Object %s", o.toString());
}
```

```java
Object o = ...; // any object
String formatter = switch(o) {
    case Integer i -> String.format("int %d", i);
    case Long l    -> String.format("long %d", l);
    case Double d  -> String.format("double %f", d);
    case Object o  -> String.format("Object %s", o.toString());
}
```

https://dev.java/learn/pattern-matching/

Programação Funcional em JavaScript

```
const originalArray = [1, 2, 3];
const newArray = [...originalArray, 4];
```

```
const person = { name: 'Alice', age: 30 };
const updatedPerson = { ...person, age: 31 };
```

```
const add = (x, y) => x + y;
const square = (x) => x * x;

function compose(...functions) {
  return (input) => functions.reduceRight((acc, fn) => fn(acc),
input);
}

const addAndSquare = compose(square, add);

console.log(addAndSquare(3, 4)); // 49
```

```
const numbers = [1, 2, 3, 4, 5, 6];

const double = (num) => num * 2;
const isEven = (num) => num % 2 === 0;

const result = numbers
  .map(double)
  .filter(isEven)
  .reduce((acc, num) => acc + num, 0);

console.log(result); // 18
```


https://hackernoon.com/functional-programming-with-javascript-a-deep-dive

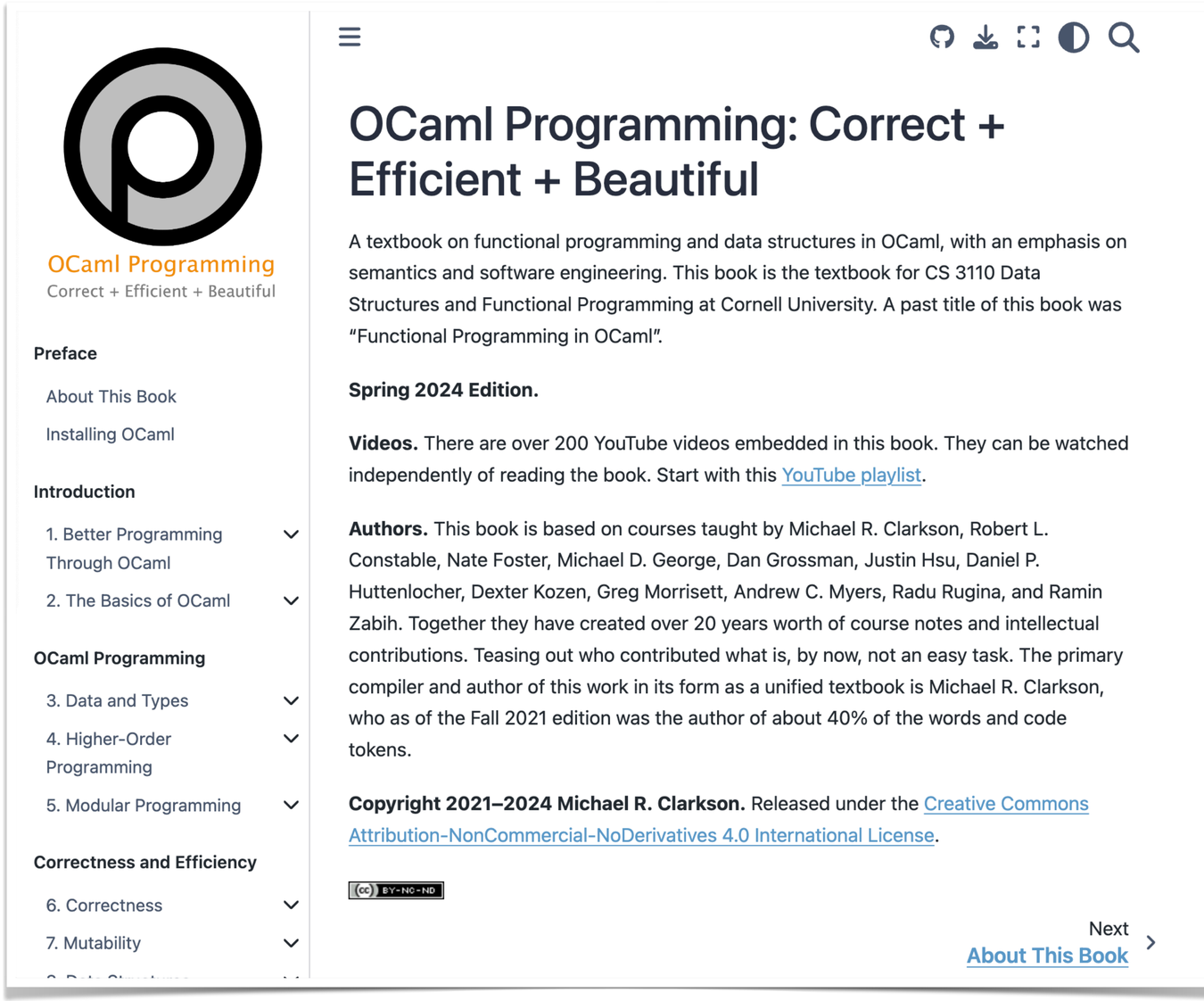
Programação Funcional (reactiva) em ReactJS

```
const BooksList = () => {

    const [books, setBooks] = useState<Book[]>([])
    const [selected, setSelected] = useState<number | undefined>(undefined)

    const [inputTitle, searchTitle, setSearchTitle] = useInput("", "Search Title")
    const [inputAuthor, searchAuthor, setSearchAuthor] = useInput("", "Search Author")

    const loadBooks = () => {
        fetch("/books.json")
            .then(response => response.json())
            .then(data => setBooks(data))

    }
    useEffect(loadBooks, [])

    const filteredBooks = books.filter(b => b.title.includes(searchTitle) && b.author.includes(searchAuthor))

    return <div>
        {inputAuthor}
        {inputTitle}
        <BooksListView books={filteredBooks} selected={selected} setSelected={setSelected} />
    </div>
}
```

Logística

Livro de texto

- OCaml Programming:
  Correct + Efficient + Beautiful
- Cornell University
- Michael R. Clarkson et al.
- Online resources
  (book, exercises, videos)



Ferramentas de laboratório

- Interpretador de OCaml + utop
- Compilador de OCaml: ocamlc + make/dune
- Trabalho com Visual Studio Code + OCaml Extension
- Exercícios com Jupyter Notebook + OCaml kernel + VSCode
- Trabalhos através do GitHub Classroom
- Correção automática dos trabalhos (mooshak ou outro equivalente)

Funcionamento

- Aulas teóricas com conceitos, discussão e exemplos.
- Aulas práticas com exercícios e apoio a projeto.

Programa

O plano para as aulas teóricas é o seguinte:

Semana	Data	Tópicos
1	6/3	Apresentação. Logística e avaliação. A história e o futuro das linguagens de programação.
1	7/3	Programação funcional. A linguagem OCaml. Expressões, Variáveis e tipos. Funções biblioteca. Input/Output básico.
1	P	Instalação das ferramentas. OCaml, Jupyter, VSCode + plugin.
2	13/3	Definição de Funções. O tipo função. Pré e pós condições. Asserções e testes unitários.
2	14/3	Funções como valores. Composição. Polimorfismo. Inferência de tipos.
2	P	Exercícios
3	20/3	Funções recursivas sobre naturais. Execução por substituição. Pensamento Indutivo vs. pensamento Iterativo.
3	21/3	Tipos estruturados: produtos e registos. Exercícios.
3	P	Exercícios
4	27/3	Tipos estruturados: tipos algébricos, pattern matching. Exercícios.
4	28/3	Férias
4	P	Exercícios
5	3/4	Tipos estruturados: Listas e funções recursivas sobre listas. Exercícios.
5	4/4	Programação de ordem superior: map e fold. Exercícios.
5	P	Exercícios. Entrega do primeiro trabalho (5/4).

Avaliação

- Componente laboratorial
  - 3 projetos (P1', P2', P3')
  - D1 - Discussão P1(30min) - 27 de Abril*
  - D2, D3 = Discussão P2, P3 (1h) - 3 de Junho*
  - Pi = min(Pi', Di)
  - CompL = 0.25 x P1 + 0.25 x P2 + 0.5 x P3, CompL >= 9.5
- Componente teórico-prática
  - 1º Teste - T1 (1h30) - 27 de Abril*
  - 2º Teste - T2 (1h30) - 3 de Junho*
  - Exame - Ex (2h30) - ?
  - CompTP = (T1 + T2) / 2 ou CompTP = Ex, CompTP >= 9.5

* datas provisórias

Nota sobre Ferramentas de Inteligência Artificial

- As ferramentas de IA é permitida nos projetos e exercícios da aula e proibida nos testes escritos e discussões.
- O uso de ferramentas de IA têm de ser explicitamente referidas no código e relatório que for entregue.
- Considera-se que um aluno que use estas ferramentas durante a realização de um exercício de avaliação ou projeto e omita que as usou comete plágio.

> “Analysis shows that, when prompted, 52 percent of ChatGPT answers to programming questions are incorrect and 77 percent are verbose. “

https://www.forbes.com/sites/garydrenik/2023/10/18/ai--low-code-can-the-two-work-harmoniously-to-democratize-coding-for-developers/

Linguagens e Ambientes de Programação (Aula Teórica 1)

LEI - Licenciatura em Engenharia Informática

João Costa Seco (joao.seco@fct.unl.pt)

NOVA SCHOOL OF
SCIENCE & TECHNOLOGY

Linguagens de Programação

Alonzo Church (1903 - 1995)

- A linguagem de programação fundamental
- O sistema formal cálculo Lambda

$$E ::= \; x \mid \; \lambda x.E \; \mid \; E \; E$$

$$(\lambda x.E) \; E' \longrightarrow E\{E'/x\} \qquad \frac{E \longrightarrow E''}{E \; E' \longrightarrow E'' \; E'}$$

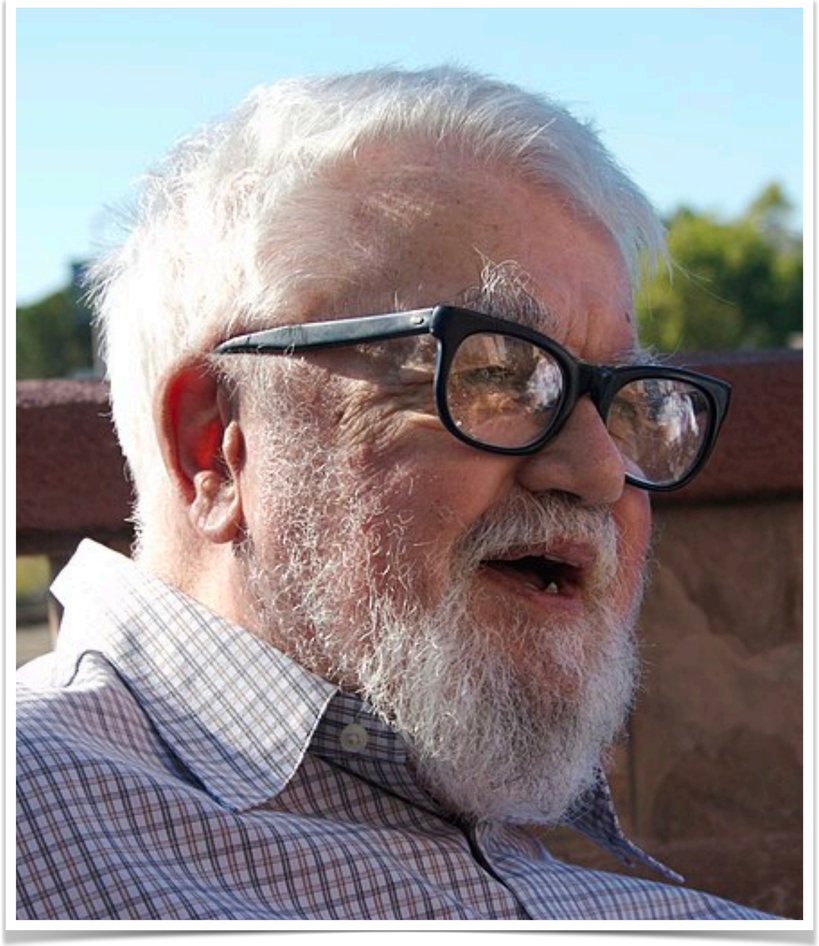
- Provou a impossibilidade de resolver o *Entscheidungsproblem* (problema de decisão)

Alan Turing (1912 - 1954)

- A máquina de Turing
- o primeiro computador (WWII)
- A indecidibilidade do “halting problem”.
- Prova que o cálculo lambda é Turing complete.

John McCarthy (1927 - 2011)

- A linguagem LISP (1960) e *garbage collection*
- LISP = LISt Processor
- Turing award 1971
- LISP já incluía :
  - Estruturas de dados em árvore
  - Garbage collection
  - Tipificação dinâmica
  - Funções de ordem superior
  - Recursão
  - Um REPL

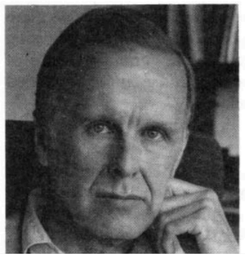


John Backus (1924 - 2007)

- Fortran (1966), BNF e FP
- Comité do ALGOL 58, e ALGOL 60
- Turing award 1977
- FP (Point Free Style) - 1977

Can Programming Be Liberated from the von Neumann Style? A Functional Style and Its Algebra of Programs

John Backus
IBM Research Laboratory, San Jose



Conventional programming languages are growing ever more enormous, but not stronger. Inherent defects at the most basic level cause them to be both fat and weak: their primitive word-at-a-time style of programming inherited from their common ancestor—the von Neumann computer, their close coupling of semantics to state transitions, their division of programming into a world of expressions and a world of statements, their inability to effectively use powerful combining forms for building new programs from existing ones, and their lack of useful mathematical properties for reasoning about programs.

O estilo Point Free não usa os nomes dos parâmetros e define funções só por composição.

```
f = (+ 1)
```

```
f x = x + 1
```

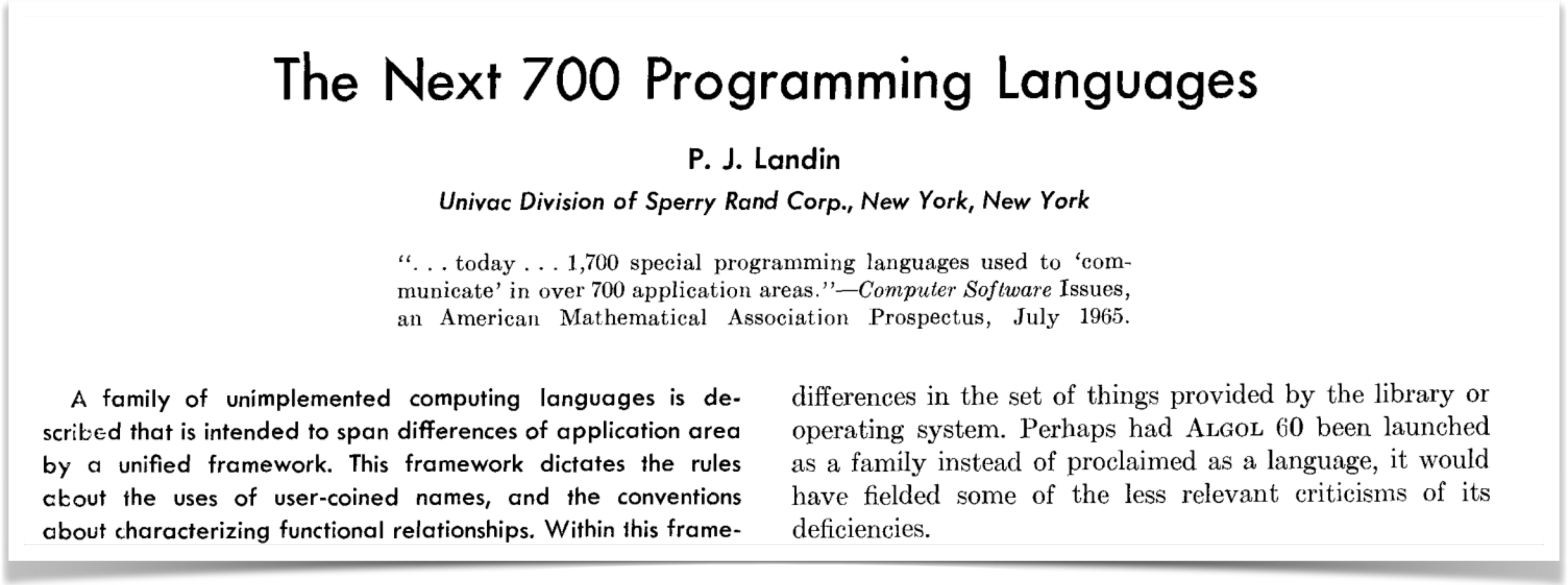
```
sum = foldr (+) 0
```

```
sum' xs = foldr (+) 0 xs
```


https://wiki.haskell.org/Pointfree

Peter Landin (1930 - 2009)

- Máquina SECD para linguagens funcionais
  - Stack, Environment, Control, Dump
  - Registos de pilhas e uma lista associativa (Environment)
  - Instruções: nil, ldc, ld, sel, join, ldf, ap, ret, dum, rap



The Next 700 Programming Languages

P. J. Landin

Univac Division of Sperry Rand Corp., New York, New York

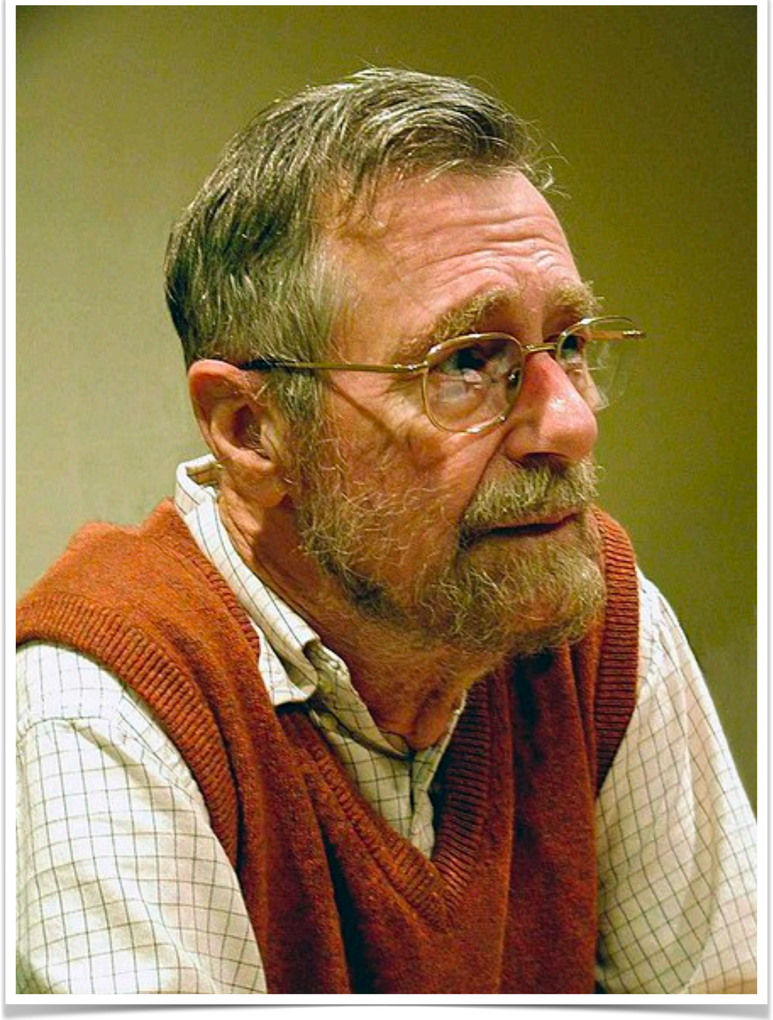
"... today ... 1,700 special programming languages used to 'communicate' in over 700 application areas."—*Computer Software* Issues, an American Mathematical Association Prospectus, July 1965.

A family of unimplemented computing languages is described that is intended to span differences of application area by a unified framework. This framework dictates the rules about the uses of user-coined names, and the conventions about characterizing functional relationships. Within this frame- differences in the set of things provided by the library or operating system. Perhaps had ALGOL 60 been launched as a family instead of proclaimed as a language, it would have fielded some of the less relevant criticisms of its deficiencies.

https://en.wikipedia.org/wiki/Peter_Landin

Edsger W. Dijkstra (1930 - 2002)

- Turing award 1972
- Definiu a noção de *structured programming*
  - Blocos de programa aninhados
  - Sem saltos
- Implementou o primeiro compilador de ALGOL



Edgar Dijkstra: Go To Statement Considered Harmful

Go To Statement Considered Harmful

Key Words and Phrases: go to statement, jump instruction, branch instruction, conditional clause, alternative clause, repetitive clause, program intelligibility, program sequencing
CR Categories: 4.22, 5.23, 5.24

EDITOR:

For a number of years I have been familiar with the observation that the quality of programmers is a decreasing function of the density of **go to** statements in the programs they produce. More recently I discovered why the use of the **go to** statement has such

dynamic progress is only characterized when we also give to which call of the procedure we refer. With the inclusion of procedures we can characterize the progress of the process via a sequence of textual indices, the length of this sequence being equal to the dynamic depth of procedure calling.

Let us now consider repetition clauses (like, **while** *B* **repeat** *A* or **repeat** *A* **until** *B*). Logically speaking, such clauses are now superfluous, because we can express repetition with the aid of recursive procedures. For reasons of realism I don't wish to exclude them: on the one hand, repetition clauses can be implemented quite comfortably with present day finite equipment; on

Tony Hoare (1934)

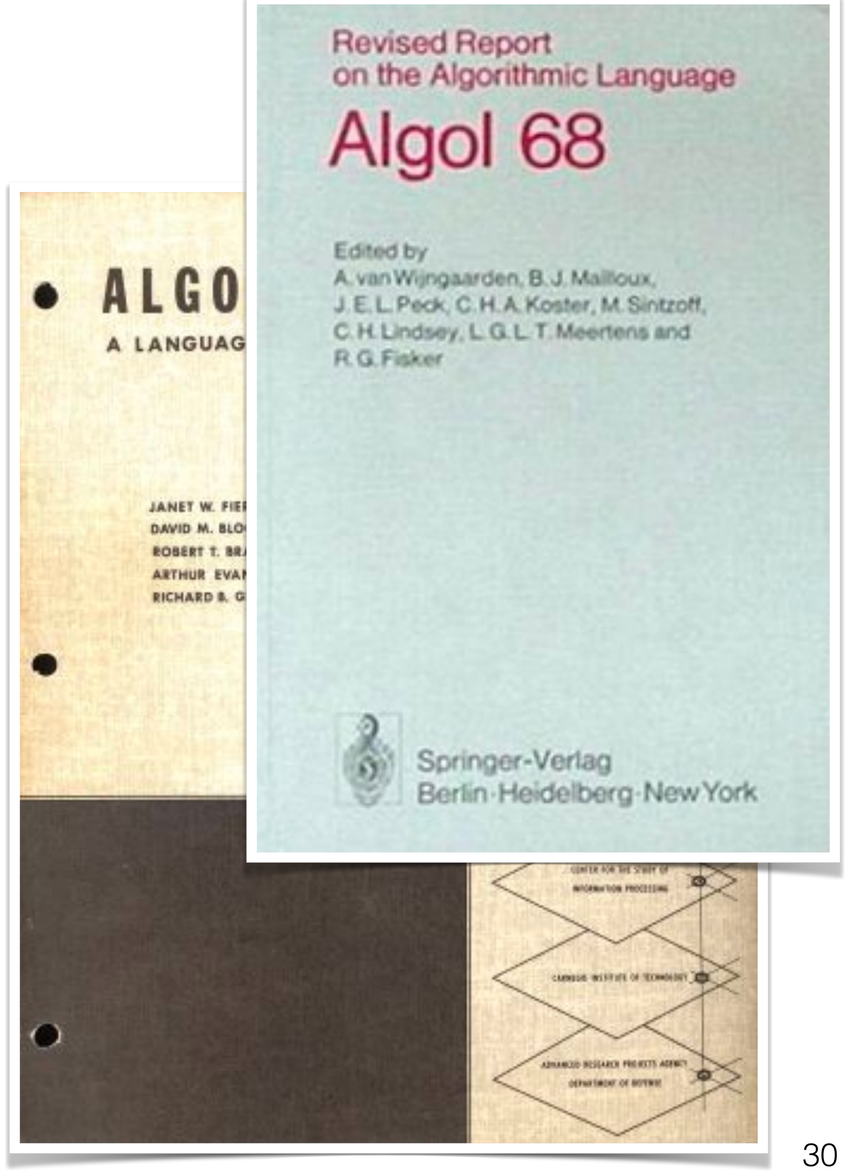
- Turing Award 1980
- Contribuições em:
  - Linguagens de Programação (ALGOL)
  - Algoritmos (Qsort)
  - Sistemas operativos (Monitors)
  - Verificação formal de programas (HL)
  - programação concorrente (CSP)
- “Null References: The Billion Dollar Mistake”

https://en.wikipedia.org/wiki/Tony_Hoare

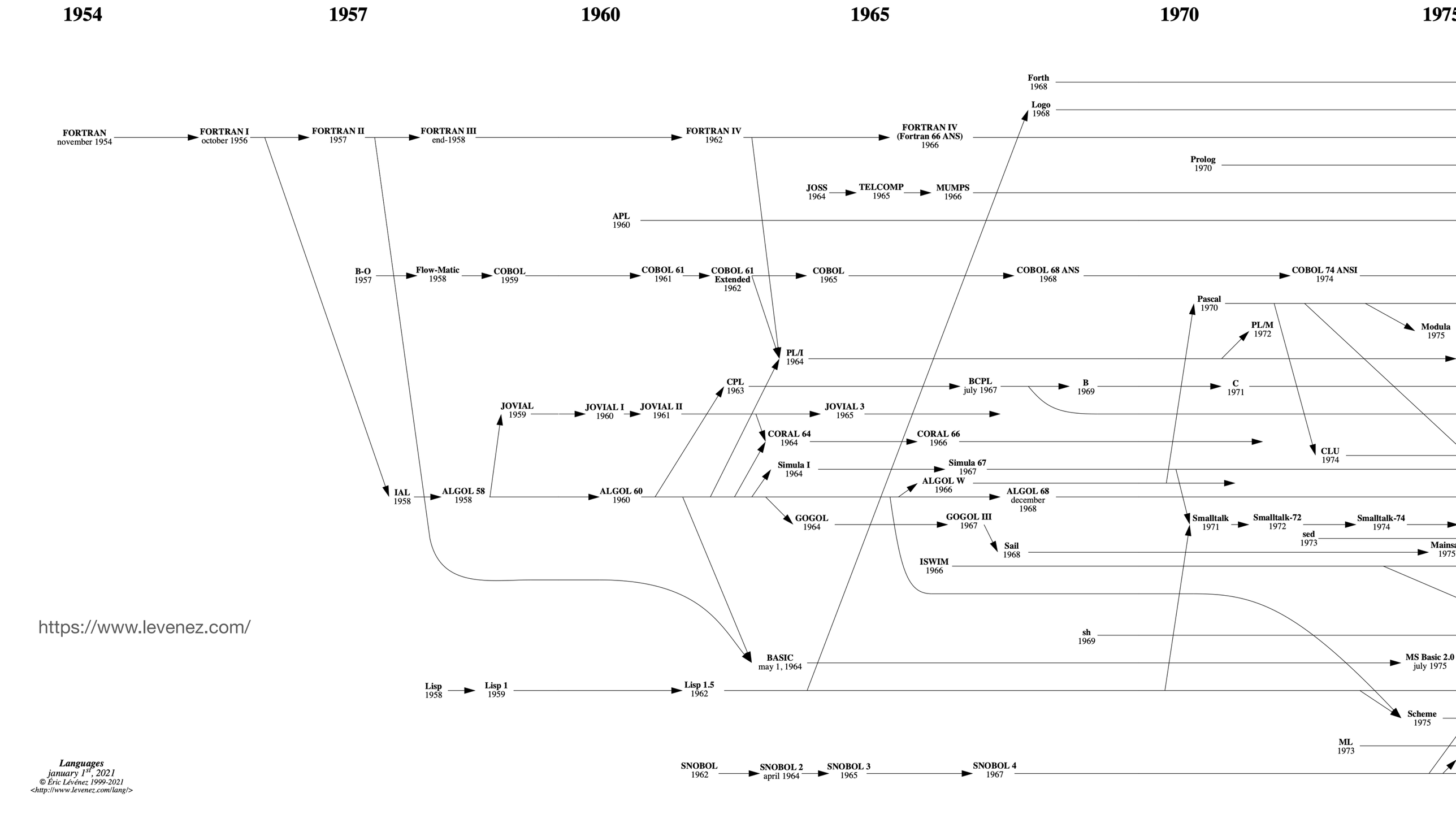
The ALGOL Family

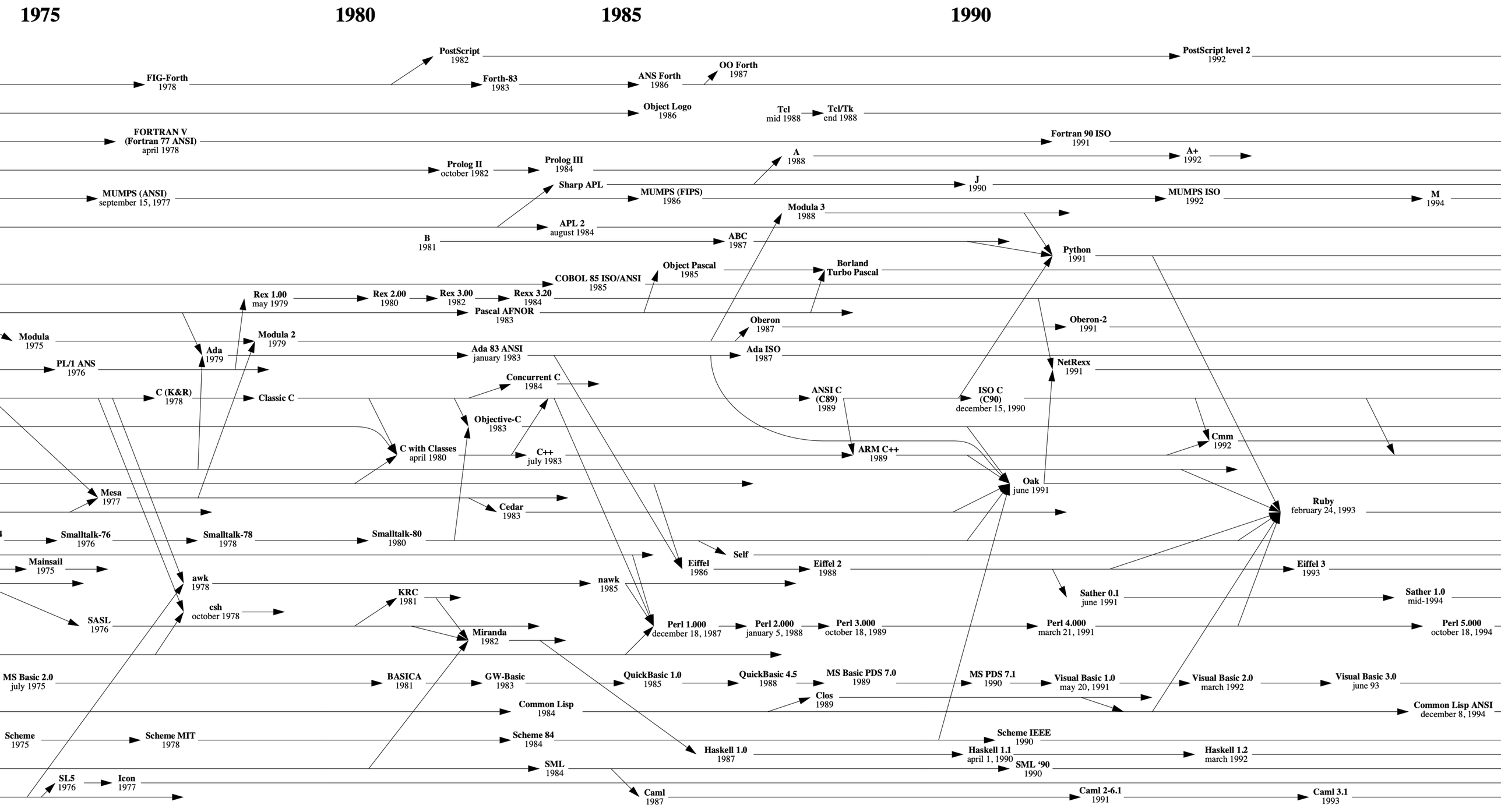
- Linguagem Imperativa concorrente com múltiplos paradigmas.
- Tipificação forte
- Objectos
- Null references

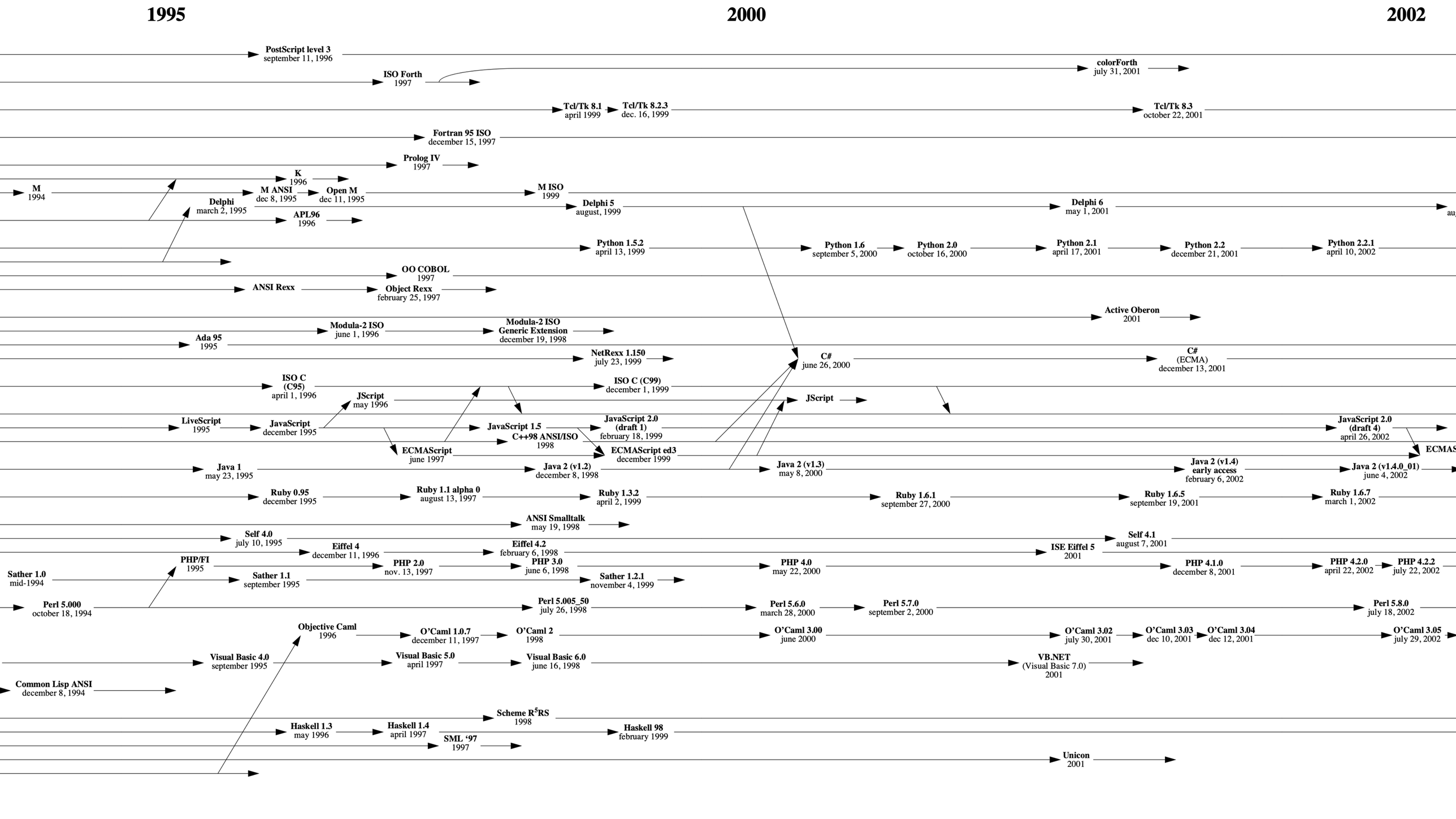
Top: John McCarthy, Fritz Bauer, Joe Wegstein. Bottom: John Backus, Peter Naur, Alan Perlis

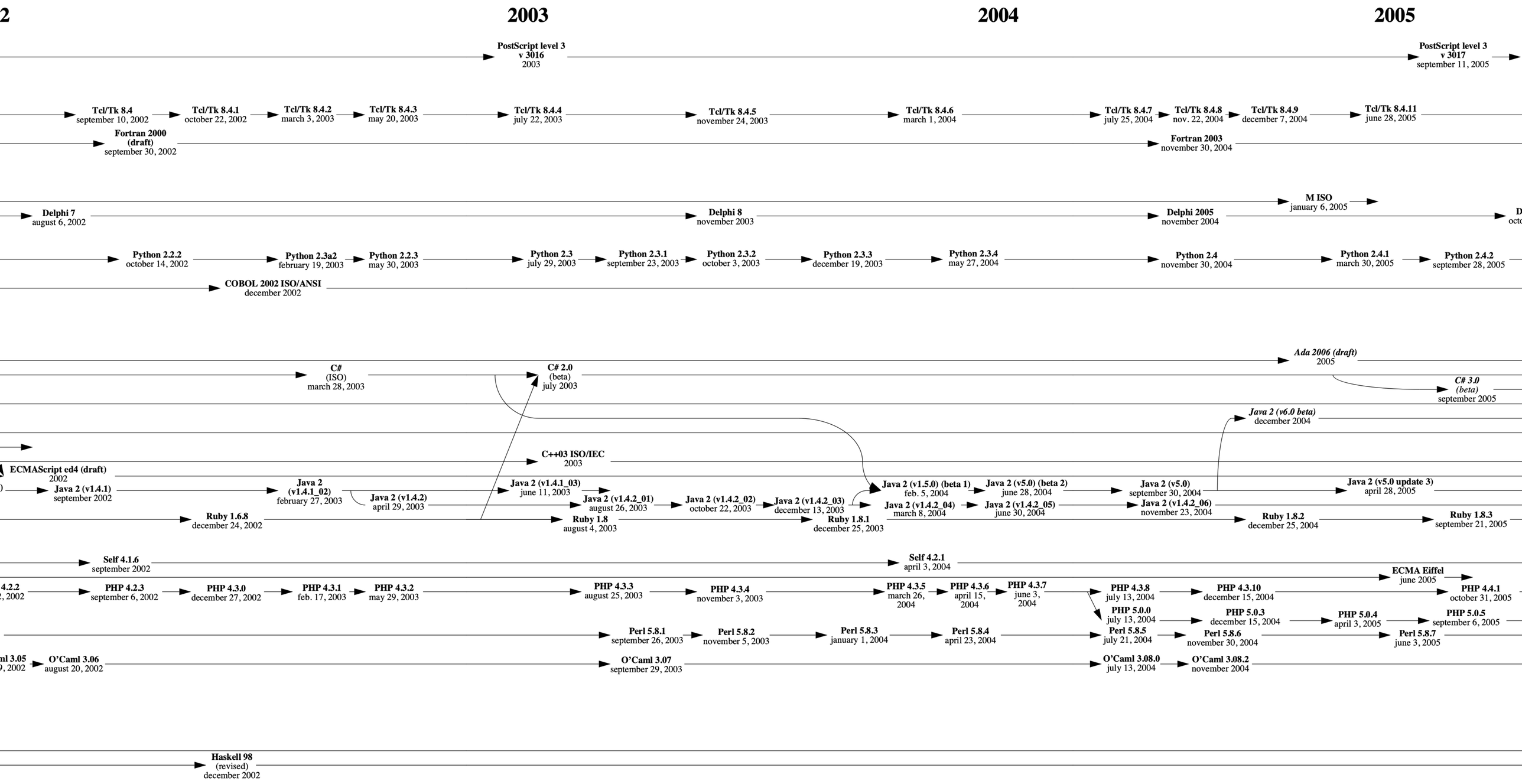


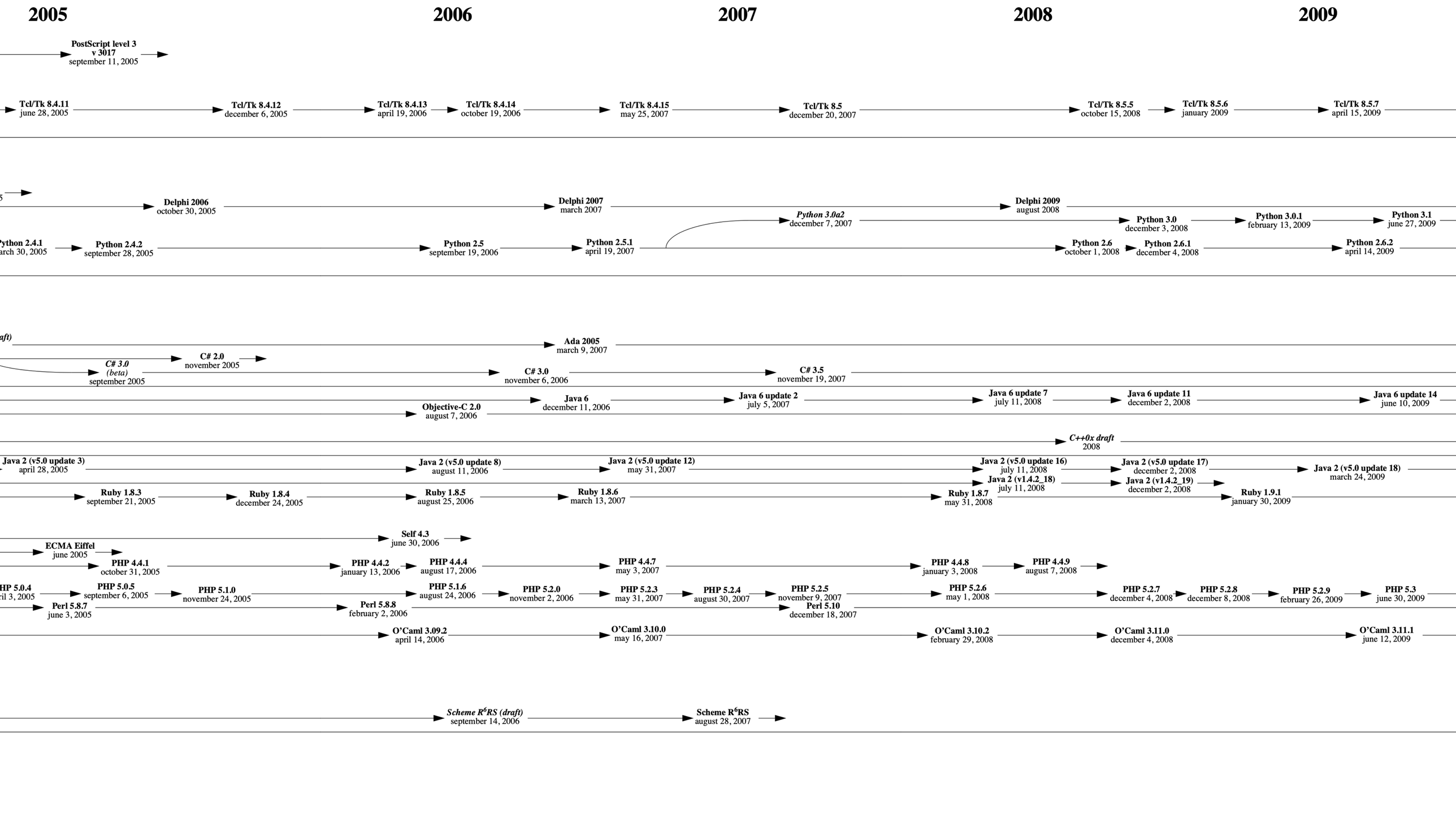
https://www-formal.stanford.edu/jmc/pictures.html

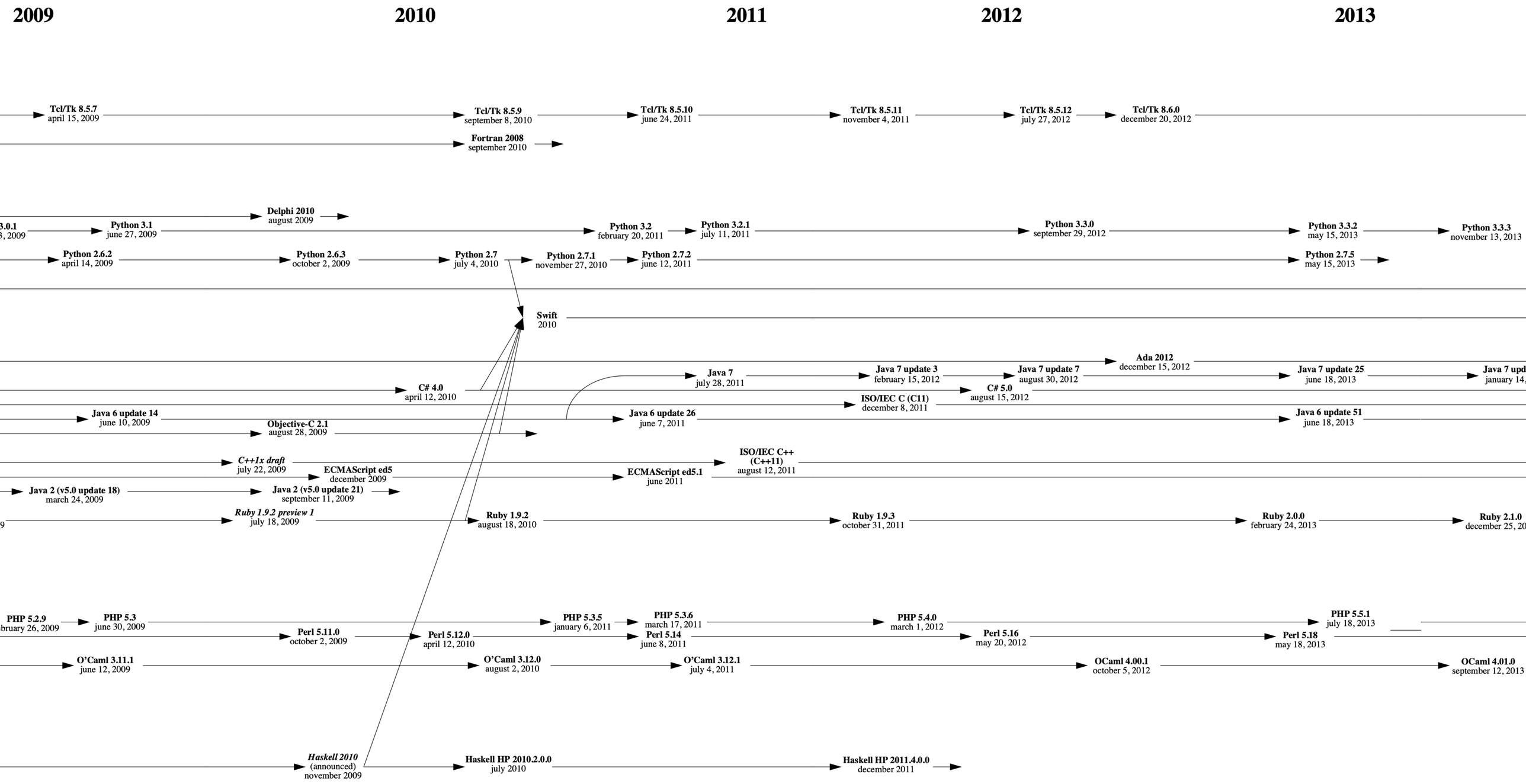

1954
1957
1960
1965
1970
1975
FORTRAN
november 1954
FORTRAN I
october 1956
FORTRAN II
1957
FORTRAN III
end-1958
FORTRAN IV
1962
FORTRAN IV
(Fortran 66 ANS)
1966
Forth
1968
Logo
1968
Prolog
1970
JOSS
1964
TELCOMP
1965
MUMPS
1966
APL
1960
B-O
1957
Flow-Matic
1958
COBOL
1959
COBOL 61
1961
COBOL 61
Extended
1962
COBOL
1965
COBOL 68 ANS
1968
COBOL 74 ANSI
1974
Pascal
1970
PL/M
1972
Modula
1975
PL/I
1964
CPL
1963
BCPL
july 1967
B
1969
C
1971
JOVIAL
1959
JOVIAL I
1960
JOVIAL II
1961
JOVIAL 3
1965
CORAL 64
1964
CORAL 66
1966
CLU
1974
Simula I
1964
Simula 67
1967
ALGOL W
1966
IAL
1958
ALGOL 58
1958
ALGOL 60
1960
ALGOL 68
december
1968
GOGOL
1964
GOGOL III
1967
Smalltalk
1971
Smalltalk-72
1972
Smalltalk-74
1974
sed
1973
Sail
1968
ISWIM
1966
sh
1969
BASIC
may 1, 1964
MS Basic 2.0
july 1975
Lisp
1958
Lisp 1
1959
Lisp 1.5
1962
Scheme
1975
ML
1973
SNOBOL
1962
SNOBOL 2
april 1964
SNOBOL 3
1965
SNOBOL 4
1967
https://www.levenez.com/
Languages
january 1st, 2021
© Éric Lévénez 1999-2021
<http://www.levenez.com/lang/>

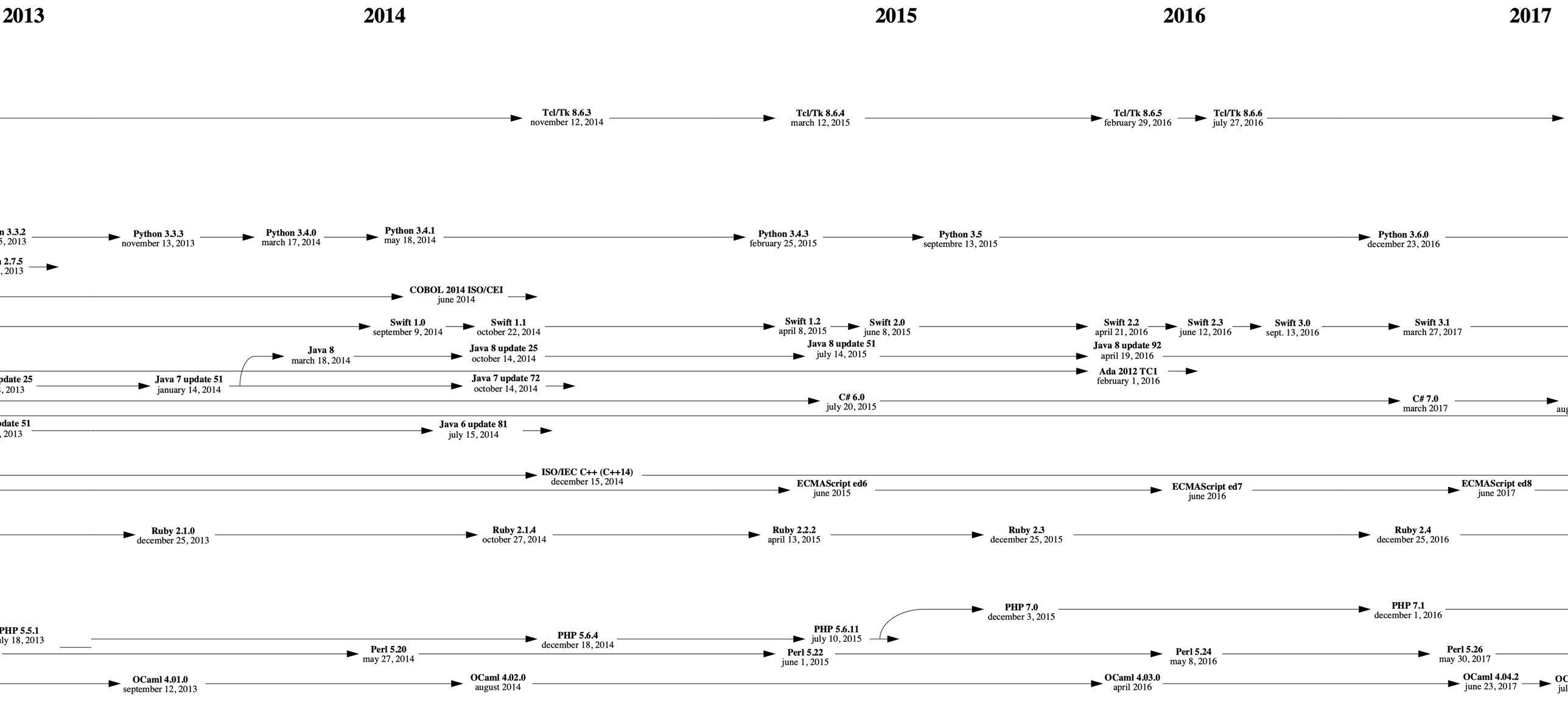

1975
1980
1985
1990
PostScript
1982
PostScript level 2
1992
FIG-Forth
1978
Forth-83
1983
ANS Forth
1986
OO Forth
1987
Object Logo
1986
Tcl
mid 1988
Tcl/Tk
end 1988
FORTRAN V
(Fortran 77 ANSI)
april 1978
Fortran 90 ISO
1991
Prolog II
october 1982
Prolog III
1984
A
1988
A+
1992
Sharp APL
J
1990
MUMPS (ANSI)
september 15, 1977
MUMPS (FIPS)
1986
MUMPS ISO
1992
M
1994
Modula 3
1988
APL 2
august 1984
B
1981
ABC
1987
Python
1991
Object Pascal
1985
Borland
Turbo Pascal
COBOL 85 ISO/ANSI
1985
Rex 1.00
may 1979
Rex 2.00
1980
Rex 3.00
1982
Rexx 3.20
1984
Pascal AFNOR
1983
Oberon
1987
Oberon-2
1991
Modula
1975
Modula 2
1979
Ada
1979
Ada 83 ANSI
january 1983
Ada ISO
1987
NetRexx
1991
PL/1 ANS
1976
Concurrent C
1984
C (K&R)
1978
Classic C
ANSI C
(C89)
1989
ISO C
(C90)
december 15, 1990
Objective-C
1983
Cmm
1992
C with Classes
april 1980
C++
july 1983
ARM C++
1989
Mesa
1977
Oak
june 1991
Cedar
1983
Ruby
february 24, 1993
Smalltalk-76
1976
Smalltalk-78
1978
Smalltalk-80
1980
Self
Mainsail
1975
Eiffel
1986
Eiffel 2
1988
Eiffel 3
1993
awk
1978
nawk
1985
Sather 0.1
june 1991
Sather 1.0
mid-1994
KRC
1981
csh
october 1978
SASL
1976
Perl 1.000
december 18, 1987
Perl 2.000
january 5, 1988
Perl 3.000
october 18, 1989
Perl 4.000
march 21, 1991
Perl 5.000
october 18, 1994
Miranda
1982
MS Basic 2.0
july 1975
BASICA
1981
GW-Basic
1983
QuickBasic 1.0
1985
QuickBasic 4.5
1988
MS Basic PDS 7.0
1989
MS PDS 7.1
1990
Visual Basic 1.0
may 20, 1991
Visual Basic 2.0
march 1992
Visual Basic 3.0
june 93
Clos
1989
Common Lisp
1984
Common Lisp ANSI
december 8, 1994
Scheme
1975
Scheme MIT
1978
Scheme 84
1984
Scheme IEEE
1990
Haskell 1.0
1987
Haskell 1.1
april 1, 1990
Haskell 1.2
march 1992
SML
1984
SML '90
1990
SL5
1976
Icon
1977
Caml
1987
Caml 2-6.1
1991
Caml 3.1
1993

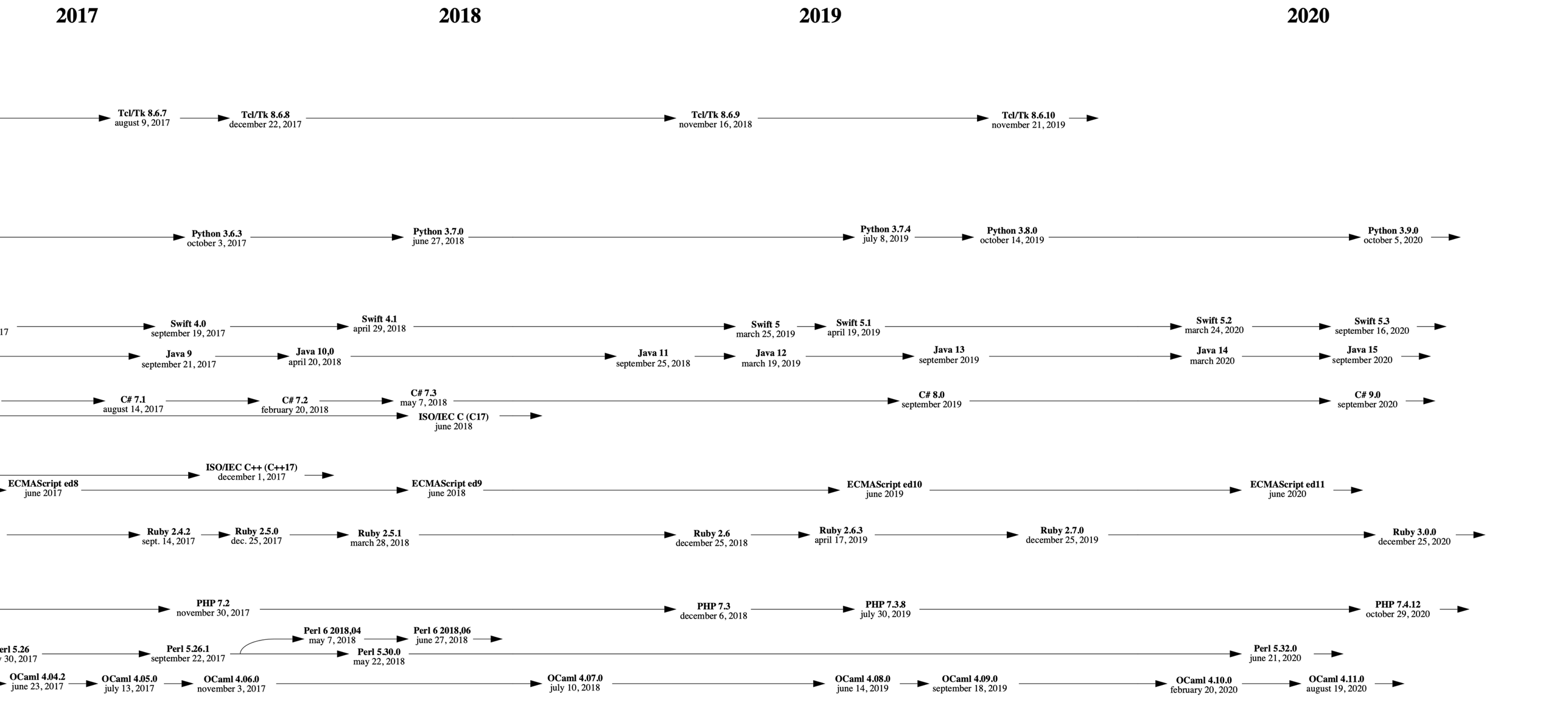

1995
2000
2002
PostScript level 3
september 11, 1996
ISO Forth
1997
colorForth
july 31, 2001
Tcl/Tk 8.1
april 1999
Tcl/Tk 8.2.3
dec. 16, 1999
Tcl/Tk 8.3
october 22, 2001
Fortran 95 ISO
december 15, 1997
Prolog IV
1997
K
1996
M
1994
M ANSI
dec 8, 1995
Open M
dec 11, 1995
M ISO
1999
Delphi
march 2, 1995
Delphi 5
august, 1999
Delphi 6
may 1, 2001
APL96
1996
Python 1.5.2
april 13, 1999
Python 1.6
september 5, 2000
Python 2.0
october 16, 2000
Python 2.1
april 17, 2001
Python 2.2
december 21, 2001
Python 2.2.1
april 10, 2002
OO COBOL
1997
ANSI Rexx
Object Rexx
february 25, 1997
Active Oberon
2001
Modula-2 ISO
june 1, 1996
Modula-2 ISO
Generic Extension
december 19, 1998
Ada 95
1995
NetRexx 1.150
july 23, 1999
C#
june 26, 2000
C#
(ECMA)
december 13, 2001
ISO C
(C95)
april 1, 1996
ISO C (C99)
december 1, 1999
JScript
may 1996
JScript
LiveScript
1995
JavaScript
december 1995
JavaScript 1.5
JavaScript 2.0
(draft 1)
february 18, 1999
JavaScript 2.0
(draft 4)
april 26, 2002
C++98 ANSI/ISO
1998
ECMAScript
june 1997
ECMAScript ed3
december 1999
Java 1
may 23, 1995
Java 2 (v1.2)
december 8, 1998
Java 2 (v1.3)
may 8, 2000
Java 2 (v1.4)
early access
february 6, 2002
Java 2 (v1.4.0_01)
june 4, 2002
Ruby 0.95
december 1995
Ruby 1.1 alpha 0
august 13, 1997
Ruby 1.3.2
april 2, 1999
Ruby 1.6.1
september 27, 2000
Ruby 1.6.5
september 19, 2001
Ruby 1.6.7
march 1, 2002
ANSI Smalltalk
may 19, 1998
Self 4.0
july 10, 1995
Self 4.1
august 7, 2001
Eiffel 4
december 11, 1996
Eiffel 4.2
february 6, 1998
ISE Eiffel 5
2001
PHP/FI
1995
PHP 2.0
nov. 13, 1997
PHP 3.0
june 6, 1998
PHP 4.0
may 22, 2000
PHP 4.1.0
december 8, 2001
PHP 4.2.0
april 22, 2002
PHP 4.2.2
july 22, 2002
Sather 1.0
mid-1994
Sather 1.1
september 1995
Sather 1.2.1
november 4, 1999
Perl 5.000
october 18, 1994
Perl 5.005_50
july 26, 1998
Perl 5.6.0
march 28, 2000
Perl 5.7.0
september 2, 2000
Perl 5.8.0
july 18, 2002
Objective Caml
1996
O'Caml 1.0.7
december 11, 1997
O'Caml 2
1998
O'Caml 3.00
june 2000
O'Caml 3.02
july 30, 2001
O'Caml 3.03
dec 10, 2001
O'Caml 3.04
dec 12, 2001
O'Caml 3.05
july 29, 2002
Visual Basic 4.0
september 1995
Visual Basic 5.0
april 1997
Visual Basic 6.0
june 16, 1998
VB.NET
(Visual Basic 7.0)
2001
Common Lisp ANSI
december 8, 1994
Scheme R5RS
1998
Haskell 1.3
may 1996
Haskell 1.4
april 1997
Haskell 98
february 1999
SML '97
1997
Unicon
2001


2
2003
2004
2005
PostScript level 3
v 3016
2003
PostScript level 3
v 3017
september 11, 2005
Tcl/Tk 8.4
september 10, 2002
Tcl/Tk 8.4.1
october 22, 2002
Tcl/Tk 8.4.2
march 3, 2003
Tcl/Tk 8.4.3
may 20, 2003
Tcl/Tk 8.4.4
july 22, 2003
Tcl/Tk 8.4.5
november 24, 2003
Tcl/Tk 8.4.6
march 1, 2004
Tcl/Tk 8.4.7
july 25, 2004
Tcl/Tk 8.4.8
nov. 22, 2004
Tcl/Tk 8.4.9
december 7, 2004
Tcl/Tk 8.4.11
june 28, 2005
Fortran 2000
(draft)
september 30, 2002
Fortran 2003
november 30, 2004
M ISO
january 6, 2005
Delphi 7
august 6, 2002
Delphi 8
november 2003
Delphi 2005
november 2004
Python 2.2.2
october 14, 2002
Python 2.3a2
february 19, 2003
Python 2.2.3
may 30, 2003
Python 2.3
july 29, 2003
Python 2.3.1
september 23, 2003
Python 2.3.2
october 3, 2003
Python 2.3.3
december 19, 2003
Python 2.3.4
may 27, 2004
Python 2.4
november 30, 2004
Python 2.4.1
march 30, 2005
Python 2.4.2
september 28, 2005
COBOL 2002 ISO/ANSI
december 2002
Ada 2006 (draft)
2005
C#
(ISO)
march 28, 2003
C# 2.0
(beta)
july 2003
C# 3.0
(beta)
september 2005
Java 2 (v6.0 beta)
december 2004
C++03 ISO/IEC
2003
ECMAScript ed4 (draft)
2002
Java 2 (v1.4.1)
september 2002
Java 2
(v1.4.1_02)
february 27, 2003
Java 2 (v1.4.2)
april 29, 2003
Java 2 (v1.4.1_03)
june 11, 2003
Java 2 (v1.4.2_01)
august 26, 2003
Java 2 (v1.4.2_02)
october 22, 2003
Java 2 (v1.4.2_03)
december 13, 2003
Java 2 (v1.5.0) (beta 1)
feb. 5, 2004
Java 2 (v1.4.2_04)
march 8, 2004
Java 2 (v5.0) (beta 2)
june 28, 2004
Java 2 (v1.4.2_05)
june 30, 2004
Java 2 (v5.0)
september 30, 2004
Java 2 (v1.4.2_06)
november 23, 2004
Java 2 (v5.0 update 3)
april 28, 2005
Ruby 1.6.8
december 24, 2002
Ruby 1.8
august 4, 2003
Ruby 1.8.1
december 25, 2003
Ruby 1.8.2
december 25, 2004
Ruby 1.8.3
september 21, 2005
Self 4.1.6
september 2002
Self 4.2.1
april 3, 2004
ECMA Eiffel
june 2005
4.2.2
2, 2002
PHP 4.2.3
september 6, 2002
PHP 4.3.0
december 27, 2002
PHP 4.3.1
feb. 17, 2003
PHP 4.3.2
may 29, 2003
PHP 4.3.3
august 25, 2003
PHP 4.3.4
november 3, 2003
PHP 4.3.5
march 26,
2004
PHP 4.3.6
april 15,
2004
PHP 4.3.7
june 3,
2004
PHP 4.3.8
july 13, 2004
PHP 4.3.10
december 15, 2004
PHP 4.4.1
october 31, 2005
PHP 5.0.0
july 13, 2004
PHP 5.0.3
december 15, 2004
PHP 5.0.4
april 3, 2005
PHP 5.0.5
september 6, 2005
Perl 5.8.1
september 26, 2003
Perl 5.8.2
november 5, 2003
Perl 5.8.3
january 1, 2004
Perl 5.8.4
april 23, 2004
Perl 5.8.5
july 21, 2004
Perl 5.8.6
november 30, 2004
Perl 5.8.7
june 3, 2005
nl 3.05
9, 2002
O'Caml 3.06
august 20, 2002
O'Caml 3.07
september 29, 2003
O'Caml 3.08.0
july 13, 2004
O'Caml 3.08.2
november 2004
Haskell 98
(revised)
december 2002


2005
2006
2007
2008
2009
PostScript level 3
v 3017
september 11, 2005
Tcl/Tk 8.4.11
june 28, 2005
Tcl/Tk 8.4.12
december 6, 2005
Tcl/Tk 8.4.13
april 19, 2006
Tcl/Tk 8.4.14
october 19, 2006
Tcl/Tk 8.4.15
may 25, 2007
Tcl/Tk 8.5
december 20, 2007
Tcl/Tk 8.5.5
october 15, 2008
Tcl/Tk 8.5.6
january 2009
Tcl/Tk 8.5.7
april 15, 2009
Delphi 2006
october 30, 2005
Delphi 2007
march 2007
Delphi 2009
august 2008
Python 2.4.2
september 28, 2005
Python 2.5
september 19, 2006
Python 2.5.1
april 19, 2007
Python 3.0a2
december 7, 2007
Python 3.0
december 3, 2008
Python 3.0.1
february 13, 2009
Python 3.1
june 27, 2009
Python 2.6
october 1, 2008
Python 2.6.1
december 4, 2008
Python 2.6.2
april 14, 2009
Ada 2005
march 9, 2007
C# 2.0
november 2005
C# 3.0
(beta)
september 2005
C# 3.0
november 6, 2006
C# 3.5
november 19, 2007
Objective-C 2.0
august 7, 2006
Java 6
december 11, 2006
Java 6 update 2
july 5, 2007
Java 6 update 7
july 11, 2008
Java 6 update 11
december 2, 2008
Java 6 update 14
june 10, 2009
C++0x draft
2008
Java 2 (v5.0 update 3)
april 28, 2005
Java 2 (v5.0 update 8)
august 11, 2006
Java 2 (v5.0 update 12)
may 31, 2007
Java 2 (v5.0 update 16)
july 11, 2008
Java 2 (v5.0 update 17)
december 2, 2008
Java 2 (v5.0 update 18)
march 24, 2009
Java 2 (v1.4.2_18)
july 11, 2008
Java 2 (v1.4.2_19)
december 2, 2008
Ruby 1.8.3
september 21, 2005
Ruby 1.8.4
december 24, 2005
Ruby 1.8.5
august 25, 2006
Ruby 1.8.6
march 13, 2007
Ruby 1.8.7
may 31, 2008
Ruby 1.9.1
january 30, 2009
Self 4.3
june 30, 2006
ECMA Eiffel
june 2005
PHP 4.4.1
october 31, 2005
PHP 4.4.2
january 13, 2006
PHP 4.4.4
august 17, 2006
PHP 4.4.7
may 3, 2007
PHP 4.4.8
january 3, 2008
PHP 4.4.9
august 7, 2008
PHP 5.0.5
september 6, 2005
PHP 5.1.0
november 24, 2005
PHP 5.1.6
august 24, 2006
PHP 5.2.0
november 2, 2006
PHP 5.2.3
may 31, 2007
PHP 5.2.4
august 30, 2007
PHP 5.2.5
november 9, 2007
PHP 5.2.6
may 1, 2008
PHP 5.2.7
december 4, 2008
PHP 5.2.8
december 8, 2008
PHP 5.2.9
february 26, 2009
PHP 5.3
june 30, 2009
Perl 5.8.7
june 3, 2005
Perl 5.8.8
february 2, 2006
Perl 5.10
december 18, 2007
O'Caml 3.09.2
april 14, 2006
O'Caml 3.10.0
may 16, 2007
O'Caml 3.10.2
february 29, 2008
O'Caml 3.11.0
december 4, 2008
O'Caml 3.11.1
june 12, 2009
Scheme R6RS (draft)
september 14, 2006
Scheme R6RS
august 28, 2007


2009
2010
2011
2012
2013
Tcl/Tk 8.5.7
april 15, 2009
Tcl/Tk 8.5.9
september 8, 2010
Tcl/Tk 8.5.10
june 24, 2011
Tcl/Tk 8.5.11
november 4, 2011
Tcl/Tk 8.5.12
july 27, 2012
Tcl/Tk 8.6.0
december 20, 2012
Fortran 2008
september 2010
Delphi 2010
august 2009
Python 3.1
june 27, 2009
Python 3.2
february 20, 2011
Python 3.2.1
july 11, 2011
Python 3.3.0
september 29, 2012
Python 3.3.2
may 15, 2013
Python 3.3.3
november 13, 2013
Python 2.6.2
april 14, 2009
Python 2.6.3
october 2, 2009
Python 2.7
july 4, 2010
Python 2.7.1
november 27, 2010
Python 2.7.2
june 12, 2011
Python 2.7.5
may 15, 2013
Swift
2010
Ada 2012
december 15, 2012
Java 7
july 28, 2011
Java 7 update 3
february 15, 2012
Java 7 update 7
august 30, 2012
Java 7 update 25
june 18, 2013
C# 4.0
april 12, 2010
C# 5.0
august 15, 2012
ISO/IEC C (C11)
december 8, 2011
Java 6 update 14
june 10, 2009
Java 6 update 26
june 7, 2011
Java 6 update 51
june 18, 2013
Objective-C 2.1
august 28, 2009
C++1x draft
july 22, 2009
ISO/IEC C++
(C++11)
august 12, 2011
ECMAScript ed5
december 2009
ECMAScript ed5.1
june 2011
Java 2 (v5.0 update 18)
march 24, 2009
Java 2 (v5.0 update 21)
september 11, 2009
Ruby 1.9.2 preview 1
july 18, 2009
Ruby 1.9.2
august 18, 2010
Ruby 1.9.3
october 31, 2011
Ruby 2.0.0
february 24, 2013
PHP 5.2.9
PHP 5.3
june 30, 2009
PHP 5.3.5
january 6, 2011
PHP 5.3.6
march 17, 2011
PHP 5.4.0
march 1, 2012
PHP 5.5.1
july 18, 2013
Perl 5.11.0
october 2, 2009
Perl 5.12.0
april 12, 2010
Perl 5.14
june 8, 2011
Perl 5.16
may 20, 2012
Perl 5.18
may 18, 2013
O'Caml 3.11.1
june 12, 2009
O'Caml 3.12.0
august 2, 2010
O'Caml 3.12.1
july 4, 2011
OCaml 4.00.1
october 5, 2012
OCaml 4.01.0
september 12, 2013
Haskell 2010
(announced)
november 2009
Haskell HP 2010.2.0.0
july 2010
Haskell HP 2011.4.0.0
december 2011


2013
2014
2015
2016
2017
Tcl/Tk 8.6.3
november 12, 2014
Tcl/Tk 8.6.4
march 12, 2015
Tcl/Tk 8.6.5
february 29, 2016
Tcl/Tk 8.6.6
july 27, 2016
Python 3.3.3
november 13, 2013
Python 3.4.0
march 17, 2014
Python 3.4.1
may 18, 2014
Python 3.4.3
february 25, 2015
Python 3.5
septembre 13, 2015
Python 3.6.0
december 23, 2016
COBOL 2014 ISO/CEI
june 2014
Swift 1.0
september 9, 2014
Swift 1.1
october 22, 2014
Swift 1.2
april 8, 2015
Swift 2.0
june 8, 2015
Swift 2.2
april 21, 2016
Swift 2.3
june 12, 2016
Swift 3.0
sept. 13, 2016
Swift 3.1
march 27, 2017
Java 8
march 18, 2014
Java 8 update 25
october 14, 2014
Java 8 update 51
july 14, 2015
Java 8 update 92
april 19, 2016
Java 7 update 51
january 14, 2014
Java 7 update 72
october 14, 2014
Ada 2012 TC1
february 1, 2016
C# 6.0
july 20, 2015
C# 7.0
march 2017
Java 6 update 81
july 15, 2014
ISO/IEC C++ (C++14)
december 15, 2014
ECMAScript ed6
june 2015
ECMAScript ed7
june 2016
ECMAScript ed8
june 2017
Ruby 2.1.0
december 25, 2013
Ruby 2.1.4
october 27, 2014
Ruby 2.2.2
april 13, 2015
Ruby 2.3
december 25, 2015
Ruby 2.4
december 25, 2016
PHP 7.0
december 3, 2015
PHP 7.1
december 1, 2016
PHP 5.6.4
december 18, 2014
PHP 5.6.11
july 10, 2015
Perl 5.20
may 27, 2014
Perl 5.22
june 1, 2015
Perl 5.24
may 8, 2016
Perl 5.26
may 30, 2017
OCaml 4.01.0
september 12, 2013
OCaml 4.02.0
august 2014
OCaml 4.03.0
april 2016
OCaml 4.04.2
june 23, 2017


2017
2018
2019
2020
Tcl/Tk 8.6.7
august 9, 2017
Tcl/Tk 8.6.8
december 22, 2017
Tcl/Tk 8.6.9
november 16, 2018
Tcl/Tk 8.6.10
november 21, 2019
Python 3.6.3
october 3, 2017
Python 3.7.0
june 27, 2018
Python 3.7.4
july 8, 2019
Python 3.8.0
october 14, 2019
Python 3.9.0
october 5, 2020
Swift 4.0
september 19, 2017
Swift 4.1
april 29, 2018
Swift 5
march 25, 2019
Swift 5.1
april 19, 2019
Swift 5.2
march 24, 2020
Swift 5.3
september 16, 2020
Java 9
september 21, 2017
Java 10,0
april 20, 2018
Java 11
september 25, 2018
Java 12
march 19, 2019
Java 13
september 2019
Java 14
march 2020
Java 15
september 2020
C# 7.1
august 14, 2017
C# 7.2
february 20, 2018
C# 7.3
may 7, 2018
C# 8.0
september 2019
C# 9.0
september 2020
ISO/IEC C (C17)
june 2018
ISO/IEC C++ (C++17)
december 1, 2017
ECMAScript ed8
june 2017
ECMAScript ed9
june 2018
ECMAScript ed10
june 2019
ECMAScript ed11
june 2020
Ruby 2.4.2
sept. 14, 2017
Ruby 2.5.0
dec. 25, 2017
Ruby 2.5.1
march 28, 2018
Ruby 2.6
december 25, 2018
Ruby 2.6.3
april 17, 2019
Ruby 2.7.0
december 25, 2019
Ruby 3.0.0
december 25, 2020
PHP 7.2
november 30, 2017
PHP 7.3
december 6, 2018
PHP 7.3.8
july 30, 2019
PHP 7.4.12
october 29, 2020
Perl 6 2018,04
may 7, 2018
Perl 6 2018,06
june 27, 2018
Perl 5.26
Perl 5.26.1
september 22, 2017
Perl 5.30.0
may 22, 2018
Perl 5.32.0
june 21, 2020
OCaml 4.04.2
june 23, 2017
OCaml 4.05.0
july 13, 2017
OCaml 4.06.0
november 3, 2017
OCaml 4.07.0
july 10, 2018
OCaml 4.08.0
june 14, 2019
OCaml 4.09.0
september 18, 2019
OCaml 4.10.0
february 20, 2020
OCaml 4.11.0
august 19, 2020

Paradigma de programação

- Imperativo (How)
  - Procedimentos
  - Objetos (classes)
  - Variáveis de estado e campos de objetos
- Declarativo (What)
  - Funções
  - Predicados (logic)
  - Sinais (reactive)
  - Eventos
  - Variáveis (no sentido matemático)