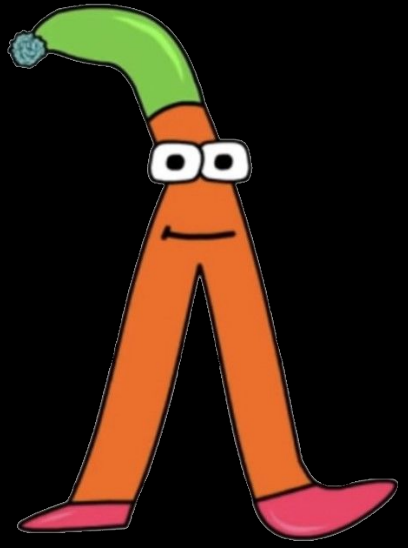
pensando funcional em um mundo de objetos



uma apresentação sutil

Guto Lanjoni

Técnico em Informática

Futuro Bacharel em Sistemas de Informação

IFSP btw

Engenheiro de Software na Nimble

Entusiasta de Open-Source

Amante de Unix

Crystal Ambassador (use Crystal)

LPI certified

OCI 5x certified

Organizador do COTESI

Escritor no dev.to nas horas vagas

Membro da :clojure-camp

Membro do Java Noroeste

Membro da **He4rt Developers**

o que é programação funcional?

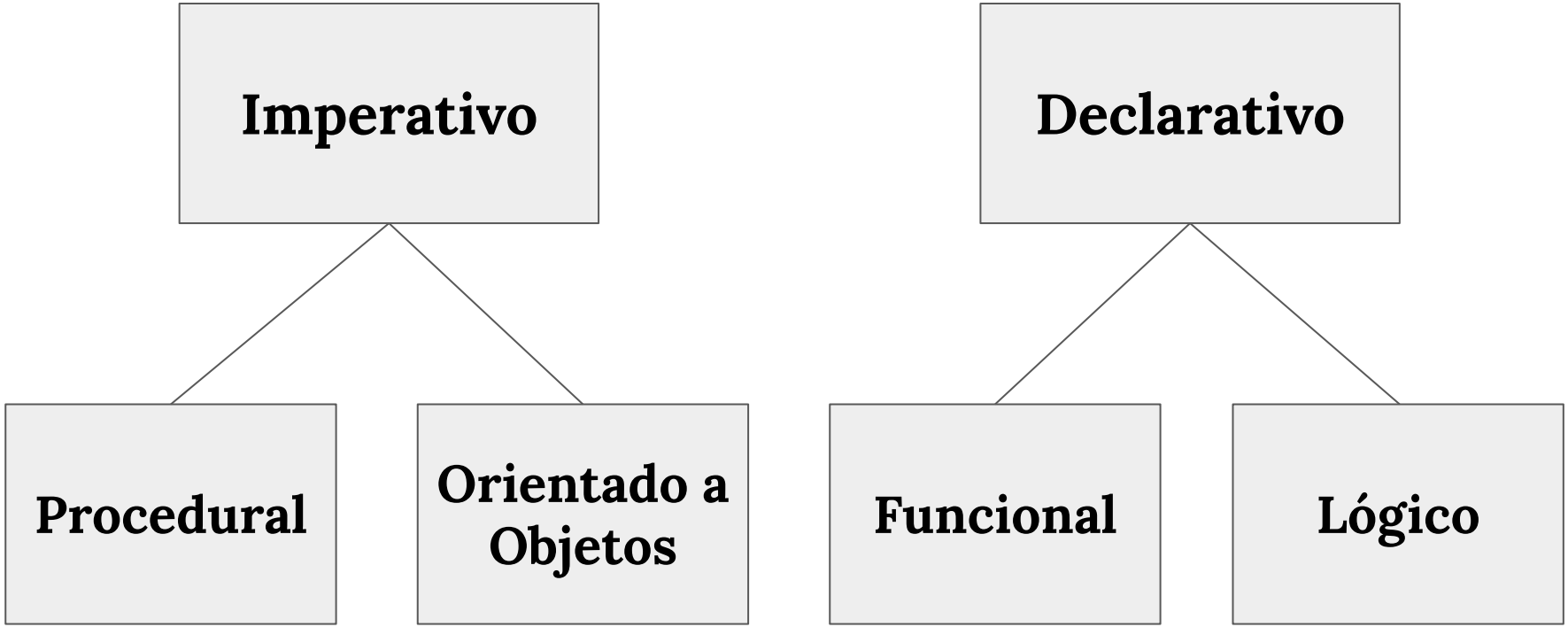
pensando com funções?

paradigma

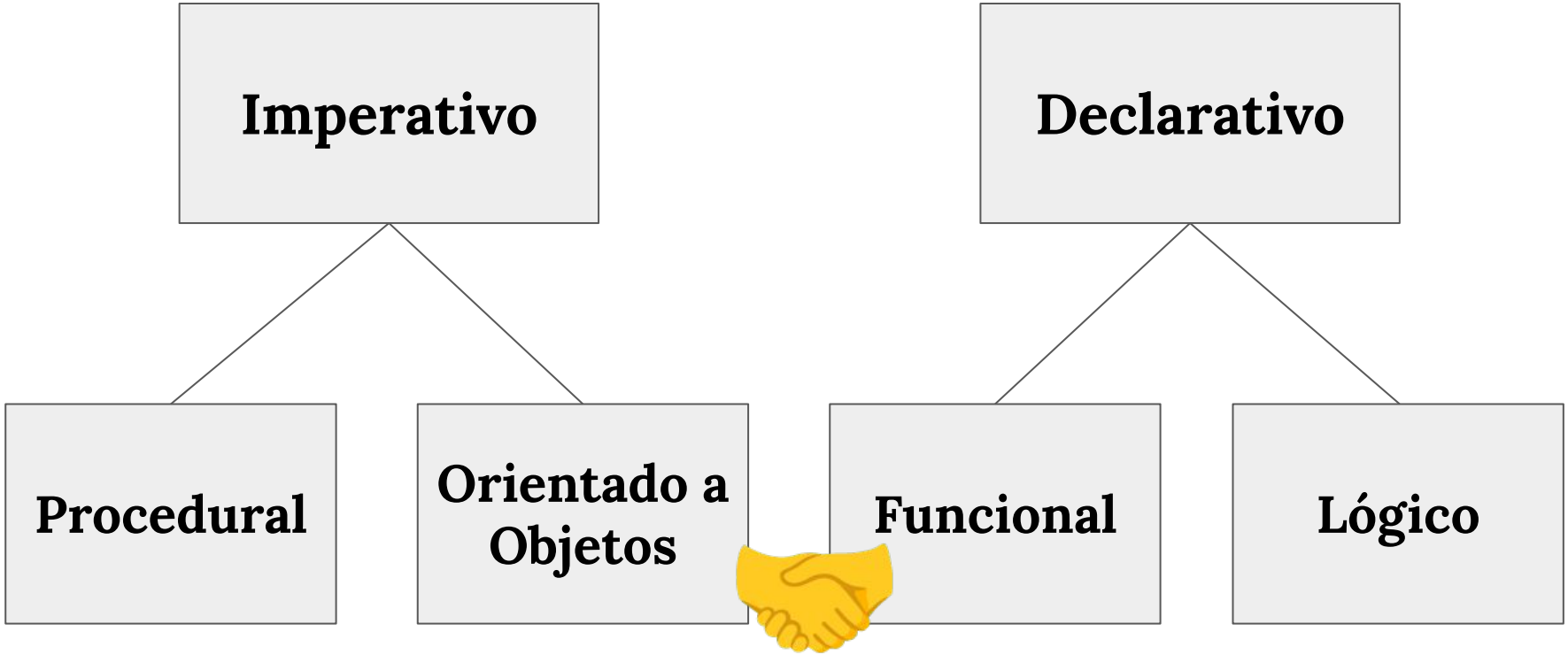
substantivo masculino

1. **um exemplo que serve como modelo; padrão.**
2. **conjunto de formas vocabulares que servem de modelo para um sistema de flexão ou de derivação (p.ex.: na declinação, na conjugação etc.); padrão.**

paradigma


Imperativo
Declarativo
Procedural
Orientado a Objetos
Funcional
Lógico

paradigma


Imperativo
Declarativo
Procedural
Orientado a Objetos
Funcional
Lógico

história

como tudo começou?

AN UNSOLVABLE PROBLEM OF ELEMENTARY NUMBER THEORY.[1]

By Alonzo Church.

Vol. 58, No. 2 (Apr., 1936), pp. 345-363 (19 pages)
https://www.jstor.org/stable/2371045

AN UNSOLVABLE PROBLEM OF ELEM
THEORY.[1]

By Alonzo Church.

Vol. 58, No. 2 (Apr., 1936), pp. 345-363 (19 pag
https://www.jstor.org/stable/2371045

Recursive Functions of Symbolic Expressions and Their Computation by Machine, Part I

John McCarthy, Massachusetts Institute of Technology, Cambridge, Mass. *

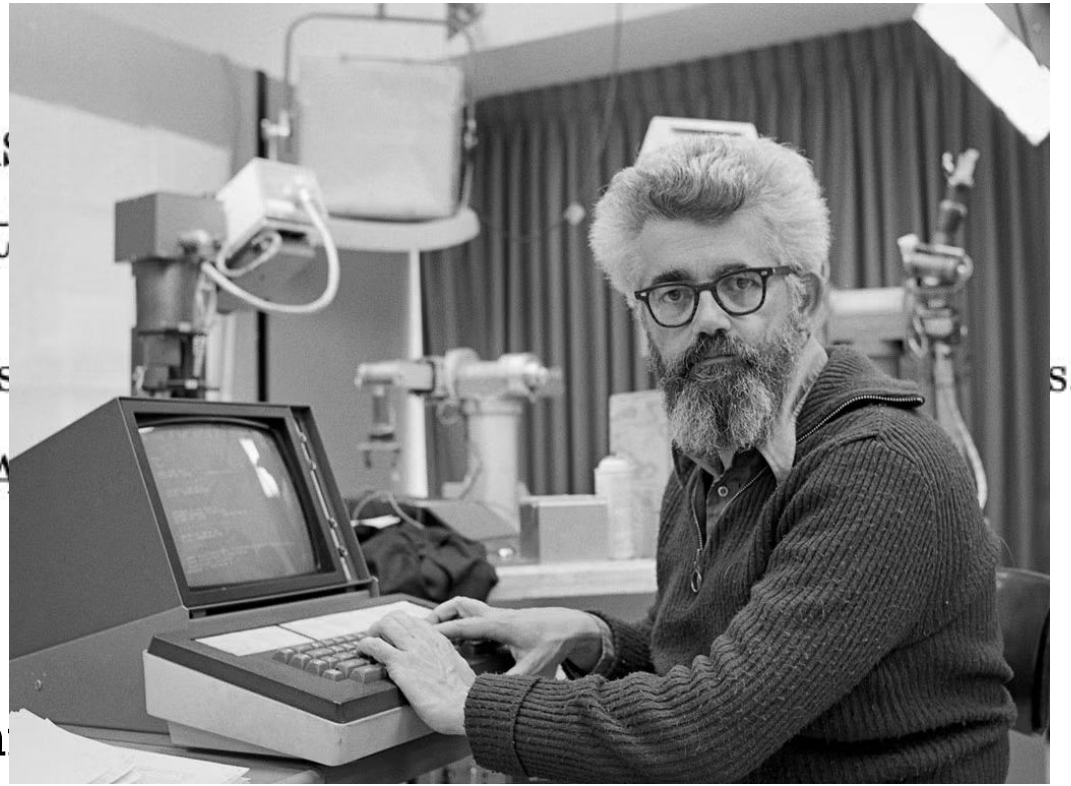
April 1960

https://www-formal.stanford.edu/jmc/recursive.pdf

Recursive Function
and Their Comput

John McCarthy, Massachus s. *

A



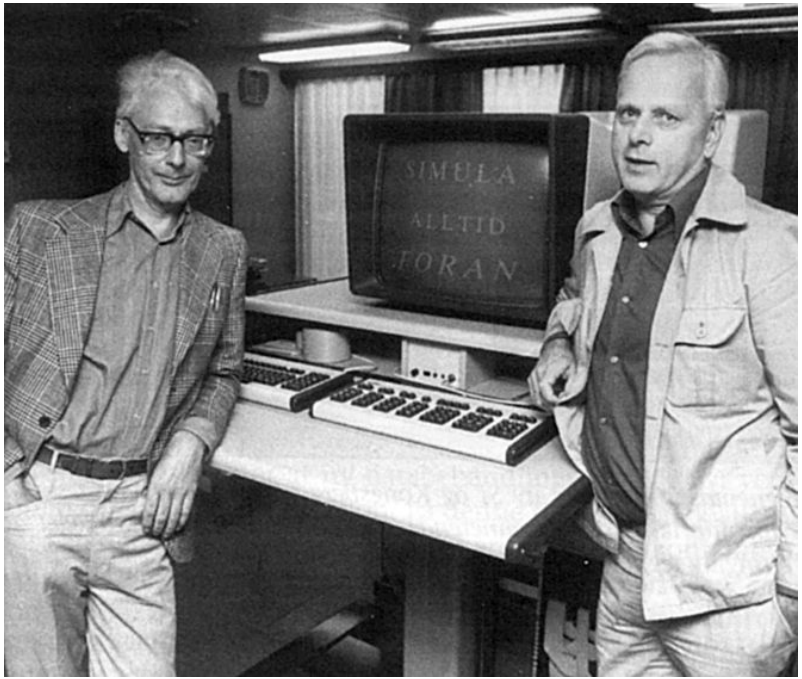
https://www-formal.stan

o que é orientação a objetos?

pensando com objetos?

SIMULA—an ALGOL-Based Simulation Language

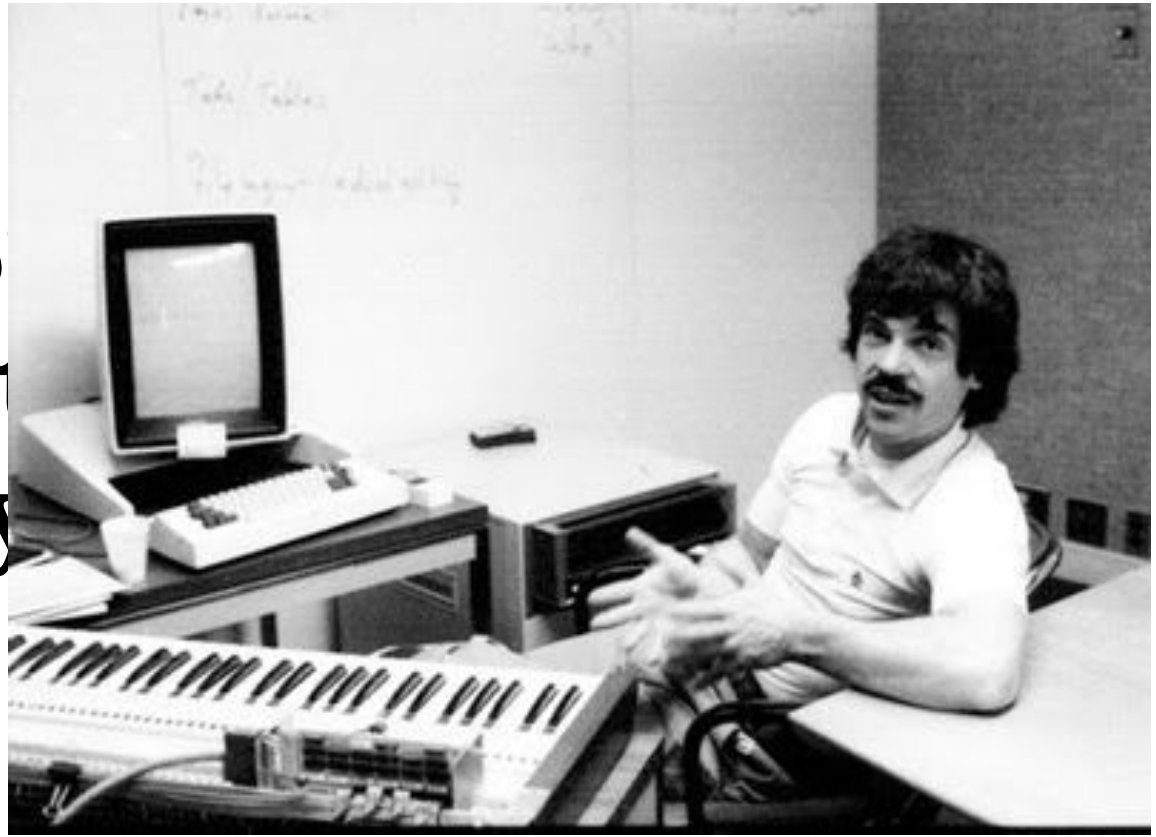
OLE-JOHAN DAHL AND KRISTEN NYGAARD
Norwegian Computing Center, Oslo, Norway



https://dl.acm.org/doi/pdf/10.1145/365813.365819 (de 1966)

os primeiros conceitos surgiram com Simula67, porém, a definição real de orientação a objetos surgiu com Alan Kay

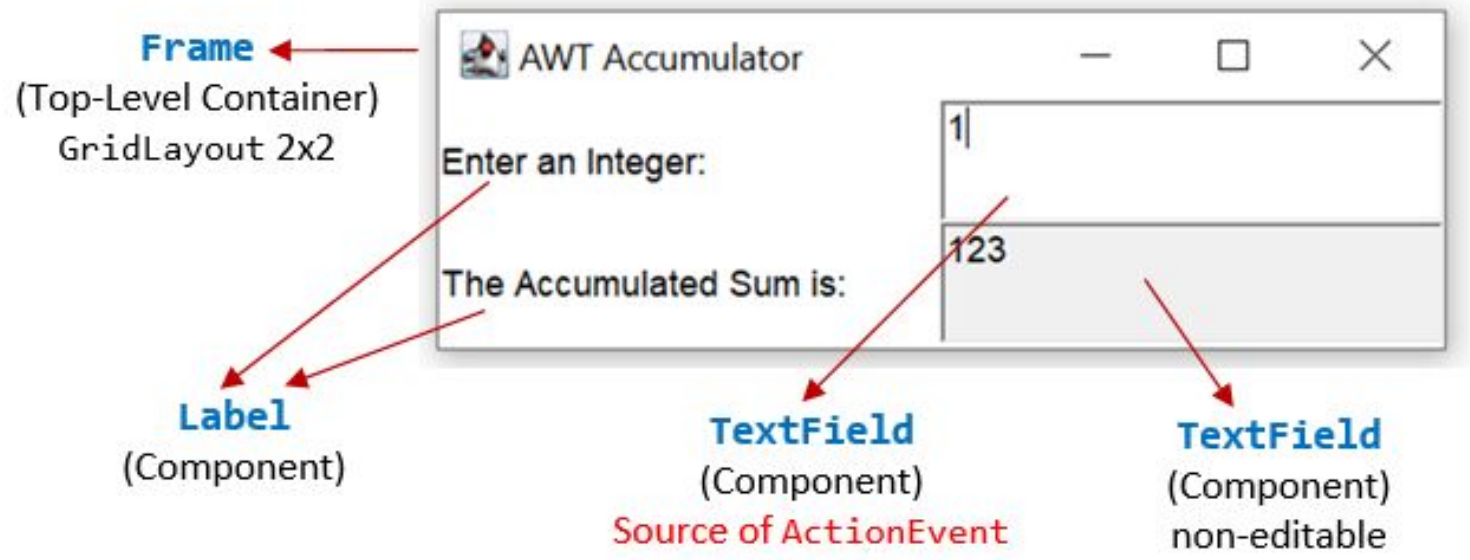
os primeiros
com Simula6
real de orient
com Alan Kay



trabalhando na Xerox, que tinha interesse em criar a primeira interface gráfica, surge então Smalltalk!


XEROX
From the Japanese computer journal "ASCII"
XEROX

cada forma da interface é compartilhada como objeto



porém a ideia principal era o *encapsulamento*

a ideia principal não era exatamente sobre classes, mas, mensagens como um todo e a troca de mensagens!

Alan Kay

Have designed a few programming languages · Upvoted by Dan Shappir, M.S. Computer... · 4y

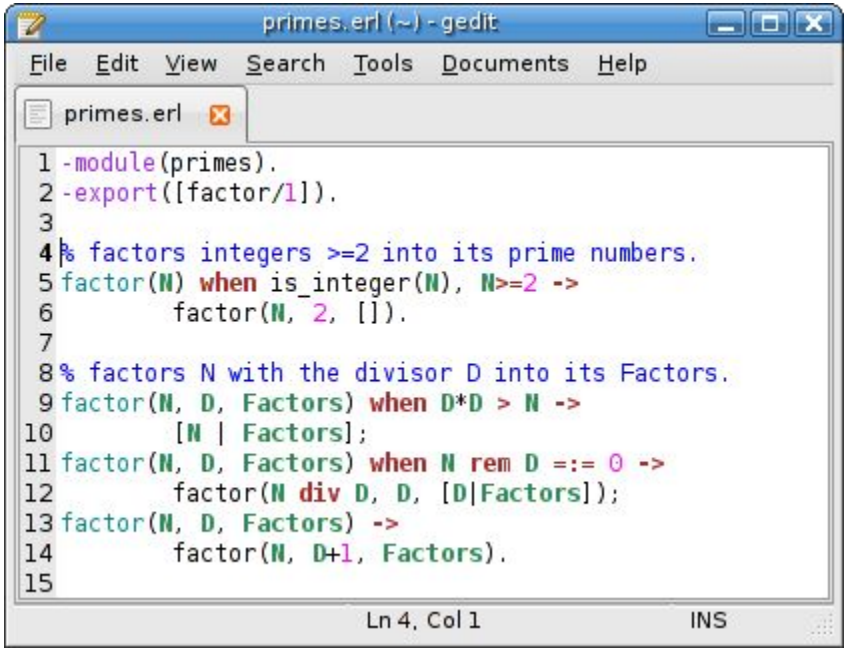
I love Joe Armstrong — we lost a great man when he recently left us.

And, he might be right. Erlang is much closer to the original ideas I had about "objects" and how to use them.

However, another way to look at this is to consider "What Is Actually Needed" (**WIAN**) and realize that much more is needed beyond what we are programming in today.

Joe would most definitely be more in favor of this idea than worrying about what either one of us did decades ago.

e Erlang é funcional.



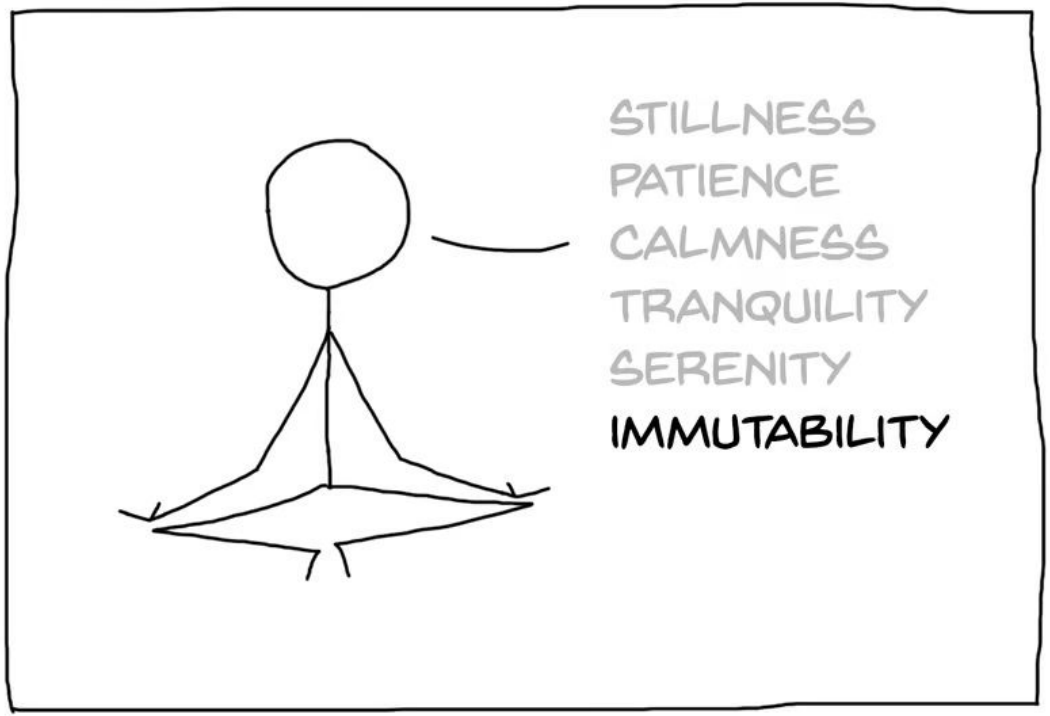
certo, mas, o que isso significa? estamos utilizando POO "errado"?

não! mas, talvez deveríamos começar a aplicar *mais conceitos funcionais* em POO...

conceitos funcionais aplicados!

quais os benefícios do paradigma funcional?

começando por:
- imutabilidade



dados imutáveis são:

- **mais seguros**
- **mais fáceis de entender**
- **mais fáceis de manipular**

assim, podemos controlar e impedir efeitos colaterais

estado

substantivo masculino

1. **conjunto de qualidades ou características com que as coisas se apresentam; conjunto de condições em que se encontram em determinado momento.**
2. **condição física de uma pessoa ou animal, ou de alguma parte de seu corpo.**

```
let a = 1;
console.log(a); // 1

a++;
console.log(a); // 2
```

```
const a = 1;
console.log(a);

a++ // erro!
```

```
let a = 1;
console.log(a); // 1

const add_one = (value) => {
  return value + 1;
}

console.log(add_one(a)); // 2
console.log(a); // 1
```

obtivemos o mesmo resultado, sem modificar o valor real!

high order functions

- são funções que recebem uma ou mais funções como parâmetros, retornando outra função

```
let a = 1;
console.log(a); // 1

const add = (increment) => {
  return (value) => {
    return value + increment;
  }
}

const add_one = add(1);
console.log(add_one(a)); // 2
console.log(add(1) (a)); // 2
console.log(a); // 1
```

mas, onde a imutabilidade pode ser aplicada no mundo real?
- toda aplicação multi-threaded

um valor imutável é thread-safe

"Immutability rocks. One of the things that are very interesting about that is: you cannot correctly represent change without immutability."

– Rich Hickey

https://www.infoq.com/presentations/The-Design-of-Datomic (53:40)

```
var values = [1, 2, 3];
var new_values = values.map((vm) => {
  if (vm == 3) {
    return 4;
  }

  return vm;
})

console.log(new_values); // [1, 2, 4]
console.log(values); // [1, 2, 3]
```

estruturas de dados persistentes

imutabilidade de forma eficiente!

criar "cópias" a todo momento pode ser tornar muito custoso ao longo do tempo...

Ideal Hash Trees

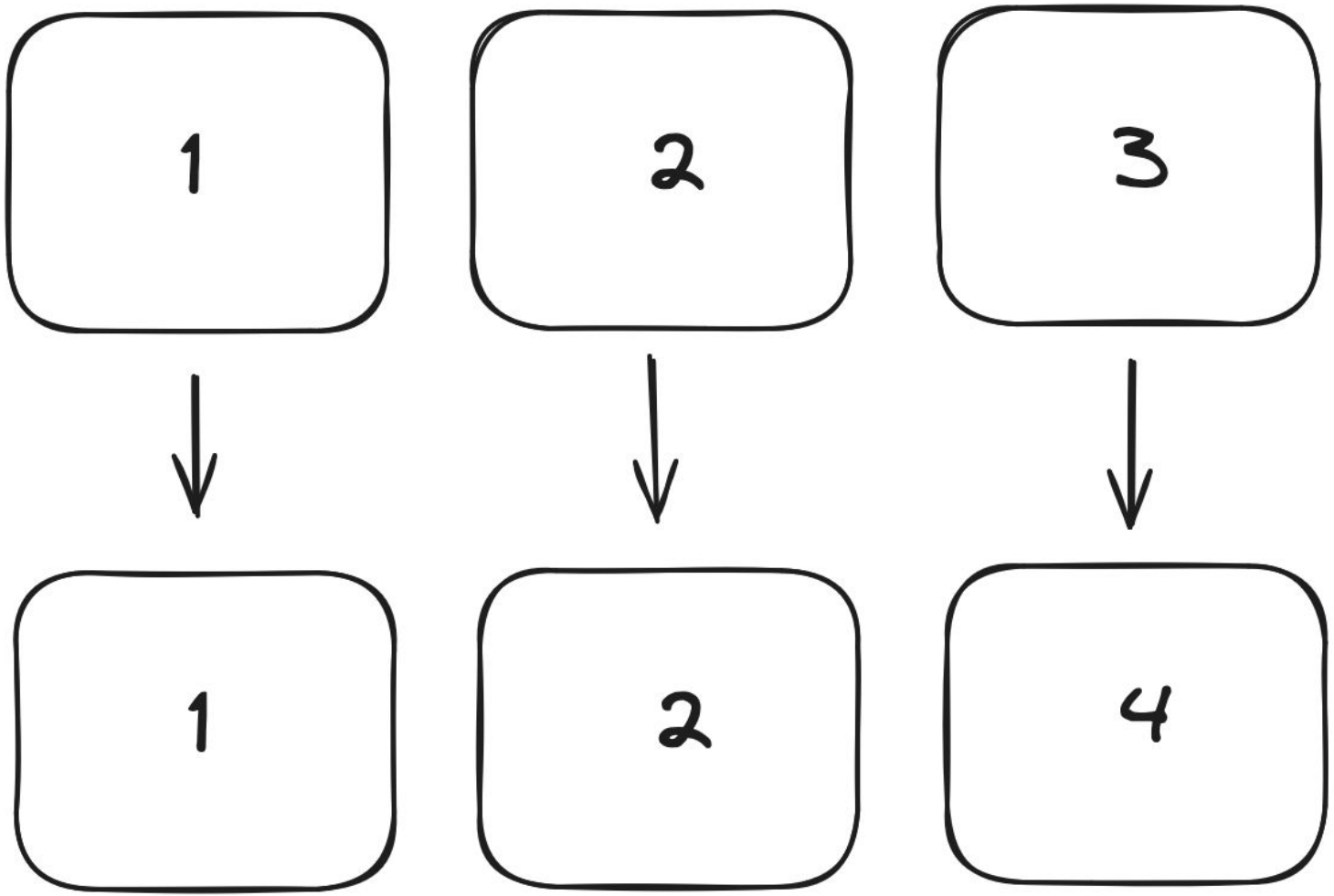
Phil Bagwell

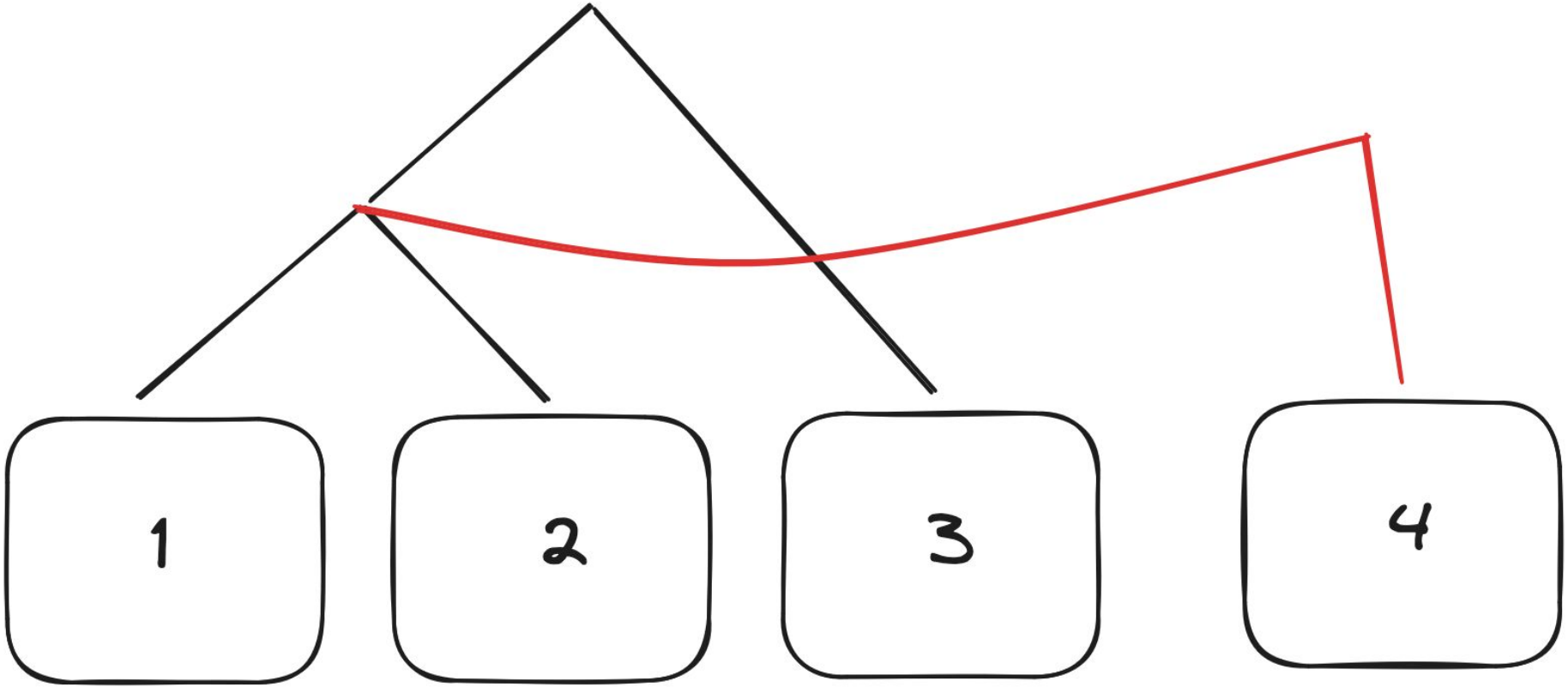
Hash Trees with nearly ideal characteristics are described. These Hash Trees require no initial root hash table yet are faster and use significantly less space than chained or double hash trees. Insert, search and delete times are small and constant, independent of key set size, operations are O(1). Small worst-case times for insert, search and removal operations can be guaranteed and misses cost less than successful searches. Array Mapped Tries(AMT), first described in Fast and Space Efficient Trie Searches, Bagwell [2000], form the underlying data structure. The concept is then applied to external disk or distributed storage to obtain an algorithm that achieves single access searches, close to single access inserts and greater than 80 percent disk block load factors. Comparisons are made with Linear Hashing, Litwin, Neimat, and Schneider [1993] and B-Trees, R.Bayer and E.M.McCreight [1972]. In addition two further applications of AMTs are briefly described, namely, Class/Selector dispatch tables and IP Routing tables. Each of the algorithms has a performance and space usage that is comparable to contemporary implementations but simpler.

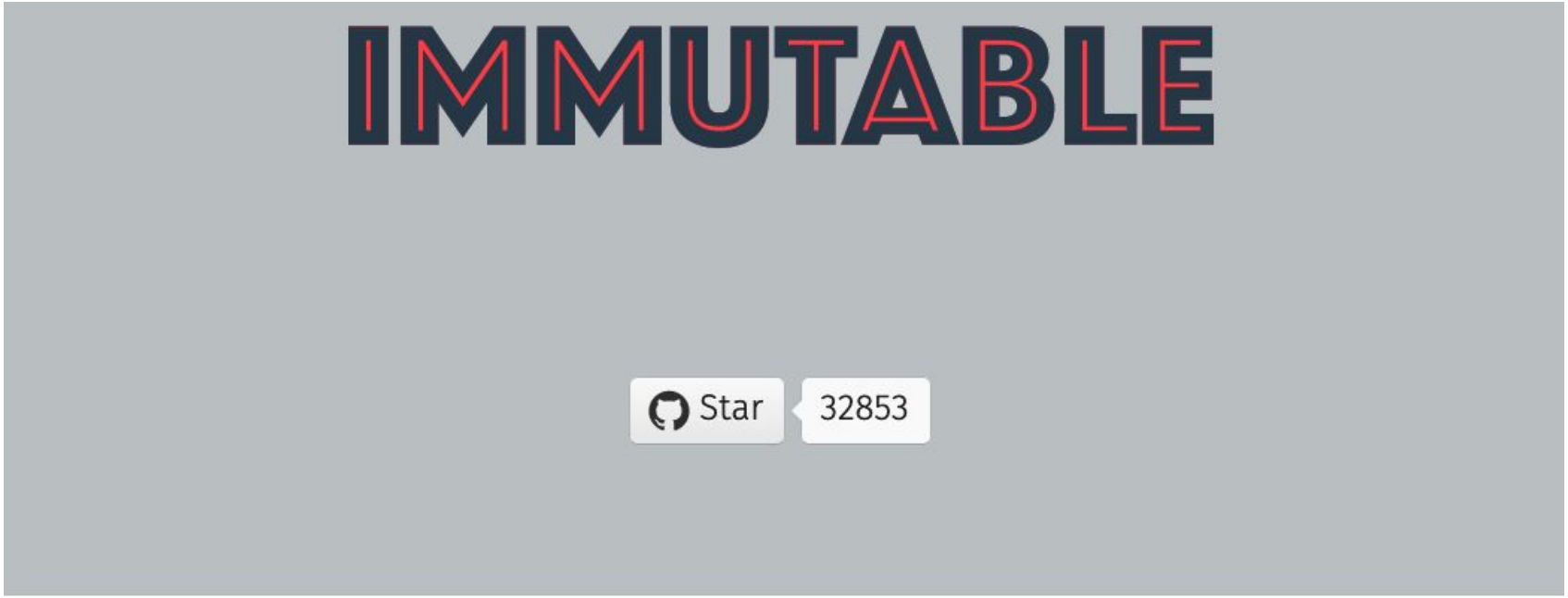
Categories and Subject Descriptors: H.4.m [**Information Systems**]: Miscellaneous

General Terms: Hashing,Hash Tables,Row Displacement,Searching,Database,Routing,Routers

https://lampwww.epfl.ch/papers/idealhashtrees.pdf


1
2
3
1
2
4


1
2
3
4



Immutable collections for JavaScript

Read the docs and eat your vegetables.

Docs are automatically generated from README.md and immutable.d.ts. Please contribute! Also, don't miss the wiki which contains articles on additional specific topics. Can't find something? Open an issue.

entendendo a complexidade

benefícios funcionais sobre manutenção

"Unlike imperative and object-oriented programming, functional programming considers everything as a function. There isn't a list of instructions or objects to be run by the computer, but rather a sequence of mathematical functions that together will solve a problem."

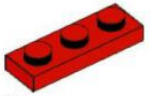
– Bruno Rodrigues

https://building.nubank.com.br/functional-programming-technology-at-nubank/

- **previsibilidade**
- **sem surpresas**
- **funções independentes?**


1x
3023, 5


1x
2412b, 5


1x
3623, 5


1x
85861, 88


8x
35480, 88


1x
4733, 88


1x
64644, 88


1x
24855, 88


6x
19119, 68


2x
98283, 150


1x
3070b, 4

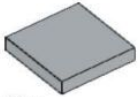

1x
15573, 4


1x
90398, 1


1x
90398, 59


1x
3005, 5


8x
98283, 86


13x
3068b, 86


2x
14719, 86


1x
63864, 86


2x
34103, 86

conclusão

finalizando esta apresentação

a programação funcional tem diversos benefícios, mas, o que nos impede de aplicar estes conceitos de diferentes formas em nossos códigos atualmente?

não é sobre programar em linguagens funcionais como um todo, mas, aplicar seus conceitos de diferentes formas, como com a imutabilidade!

@gutolanjoni no twitter
@gutolanjoni no insta
@gutolanjoni no telegram
@lanjoni no github
dev.to/guto
lanjoni.dev
twitch.tv/gutolanjoni