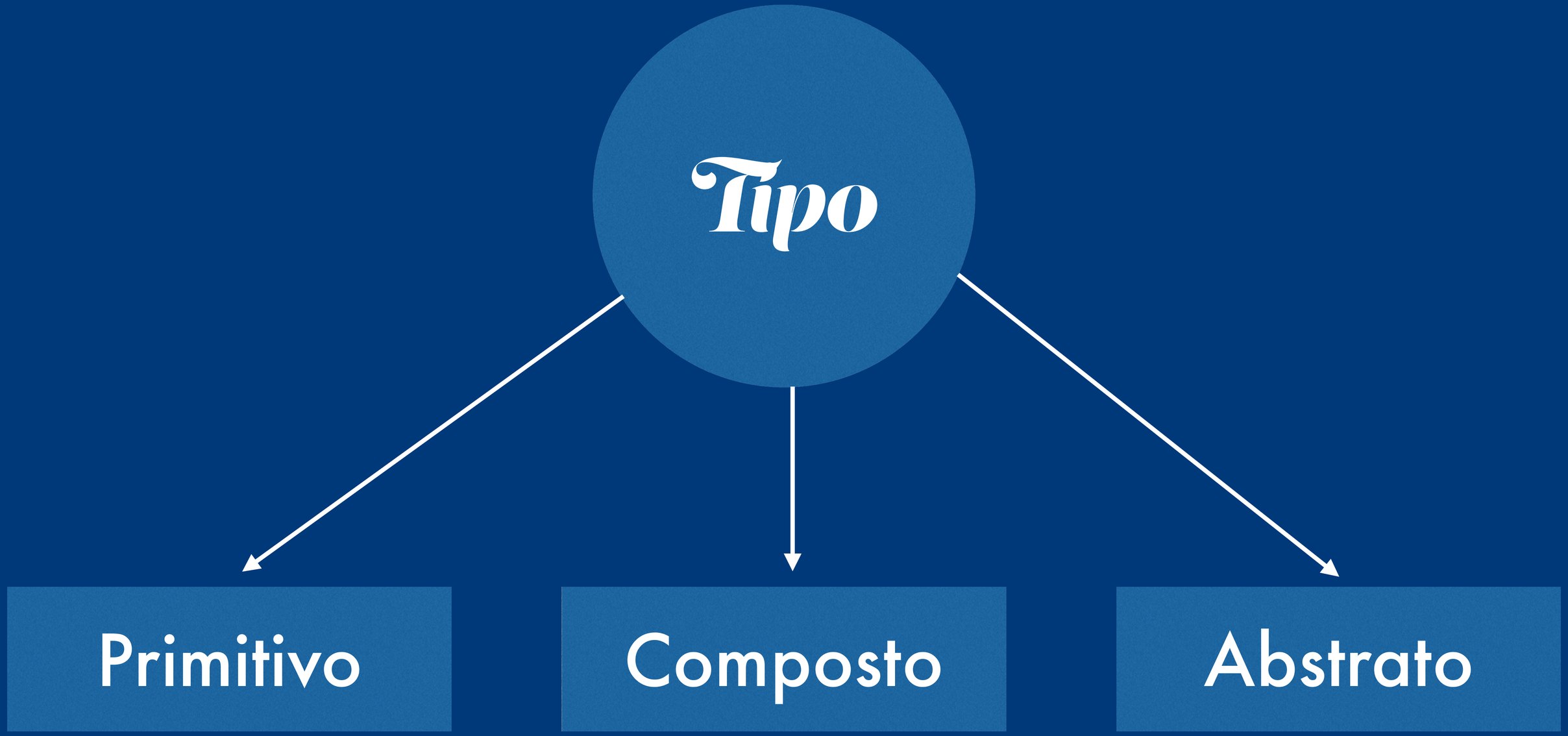
Sistemas de Tipos Modernos

O que é um sistema de tipos?

~~O que é um sistema de tipos?~~

O que é um tipo?


Tipo


Tipo
Primitivo
Composto
Abstrato

Primitivo

Short

Integer

Byte

Double

Float

Decimal

Boolean

Bignum

Composto

Object

Array

Union

Struct

Enumeration

String

Tuple

Tagged Union

Abstrato

Stack

Graph

Queue

Tree

Hash/Map/Dict

List

Object

O que é um sistema de tipos?

Estático

versus

Dinâmico

Estático

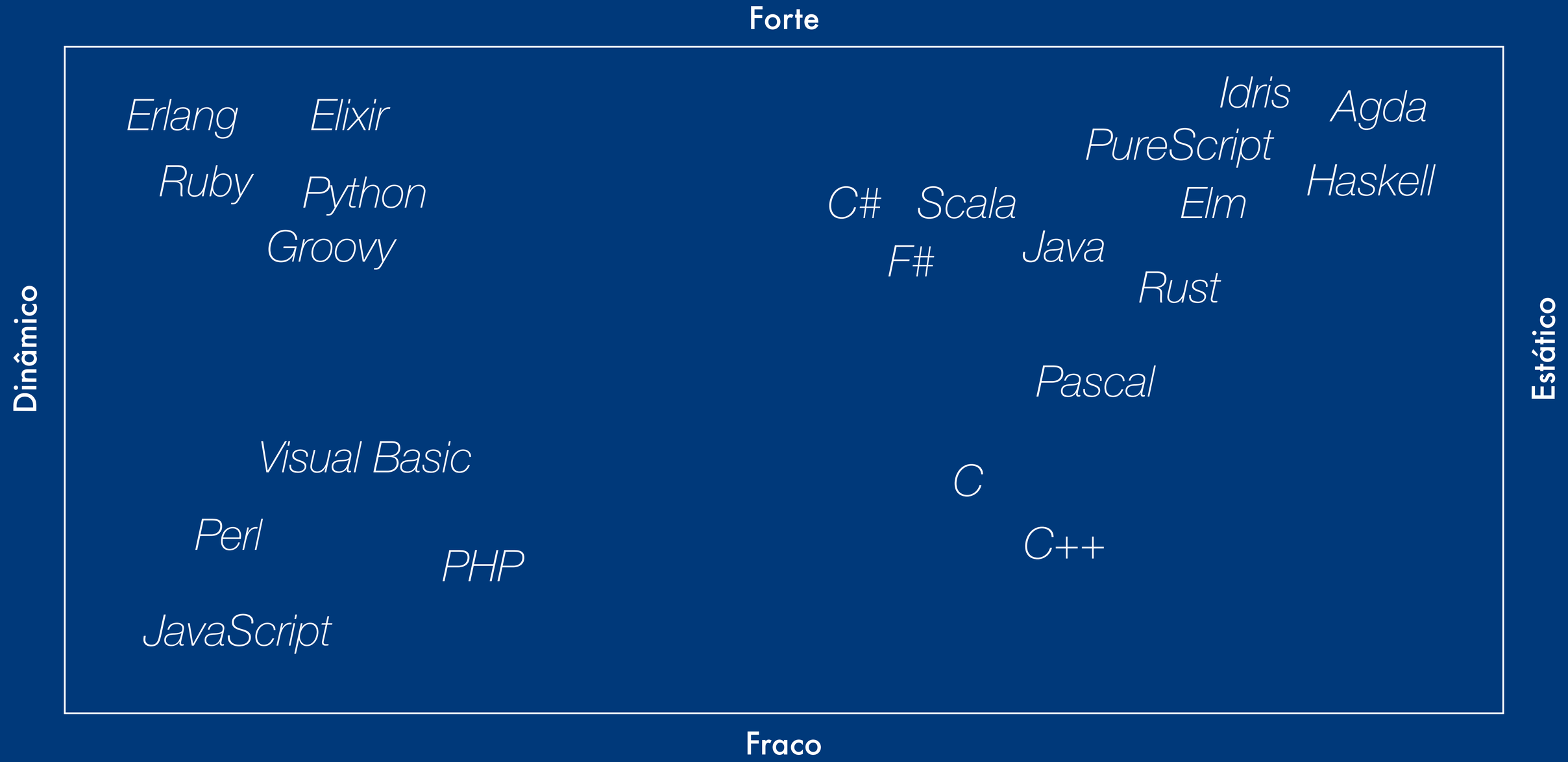
e também

Dinâmico

Forte

versus

Fraco

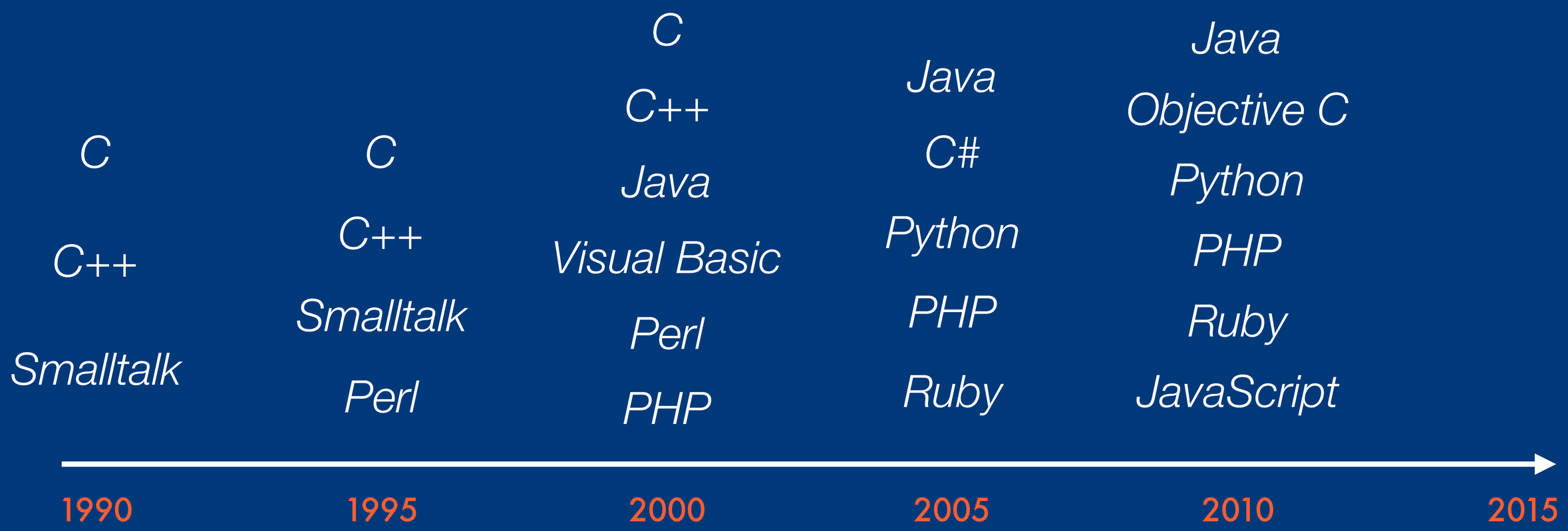

Forte
Dinâmico
Estático
Fraco
Erlang
Elixir
Ruby
Python
Groovy
Idris
Agda
PureScript
Haskell
C#
Scala
Elm
F#
Java
Rust
Pascal
Visual Basic
C
Perl
C++
PHP
JavaScript

Implícito

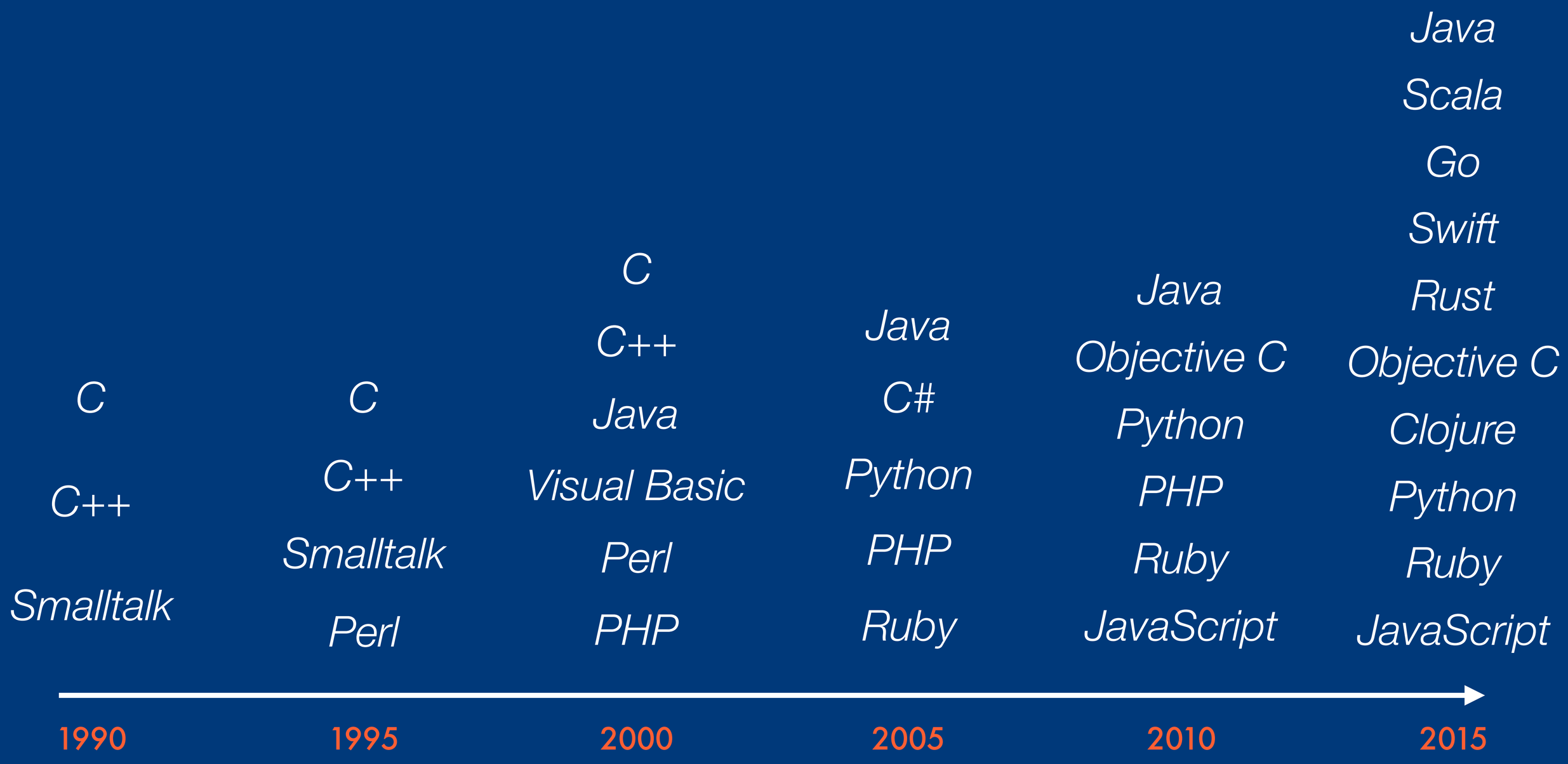
versus

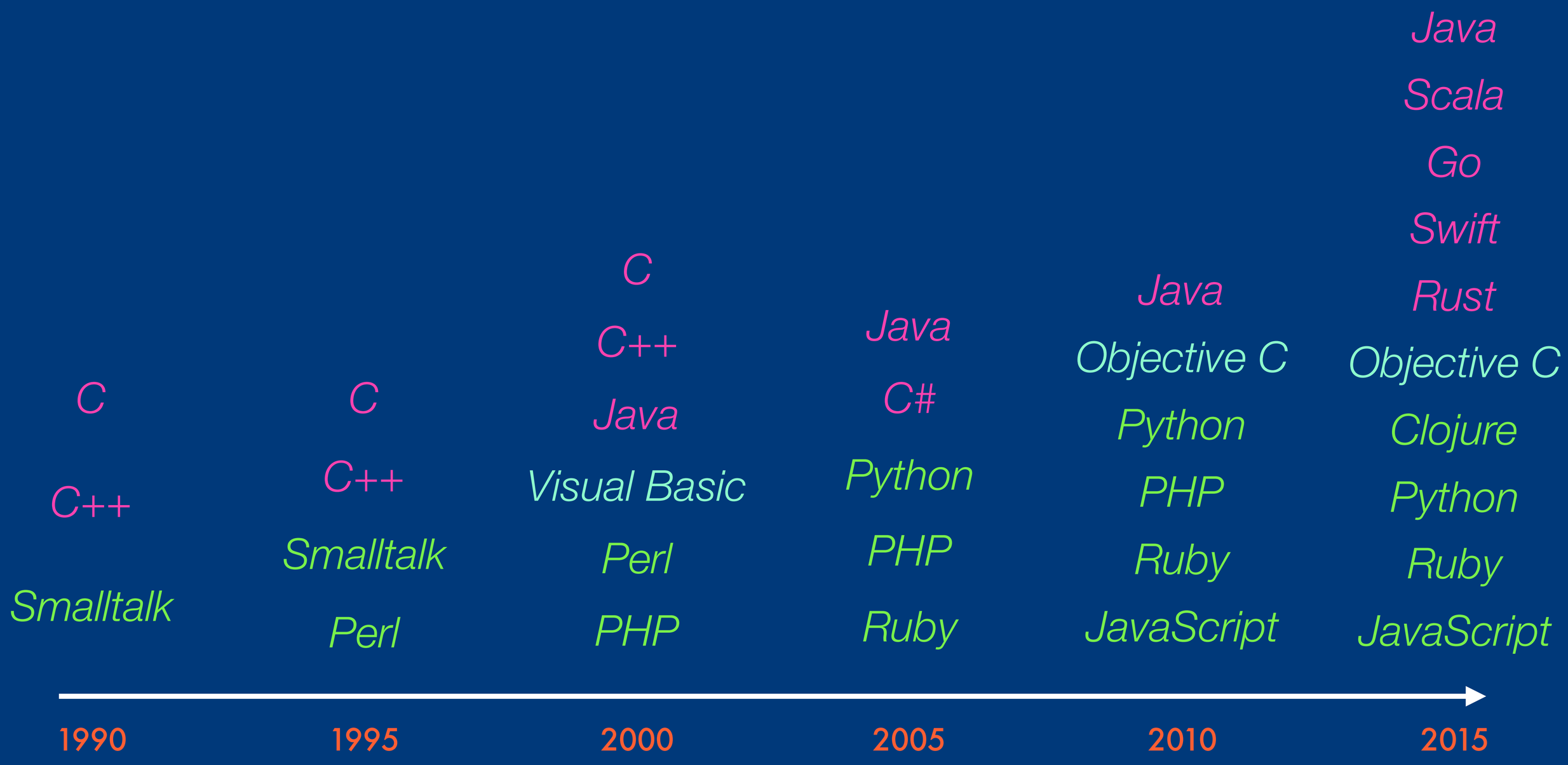
Explícito

A Hora e a Vez dos Sistemas Estáticos



Principais linguagens na indústria


C
C++
Smalltalk
C
C++
Smalltalk
Perl
C
C++
Java
Visual Basic
Perl
PHP
Java
C#
Python
PHP
Ruby
Java
Objective C
Python
PHP
Ruby
JavaScript
Java
Scala
Go
Swift
Rust
Objective C
Clojure
Python
Ruby
JavaScript
1990
1995
2000
2005
2010
2015


C
C++
Smalltalk
C
C++
Smalltalk
Perl
C
C++
Java
Visual Basic
Perl
PHP
Java
C#
Python
PHP
Ruby
Java
Objective C
Python
PHP
Ruby
JavaScript
Java
Scala
Go
Swift
Rust
Objective C
Clojure
Python
Ruby
JavaScript
1990
1995
2000
2005
2010
2015

O que aconteceu?

Dinâmico

Estático

Dinâmico

Estático

No prato dinâmico

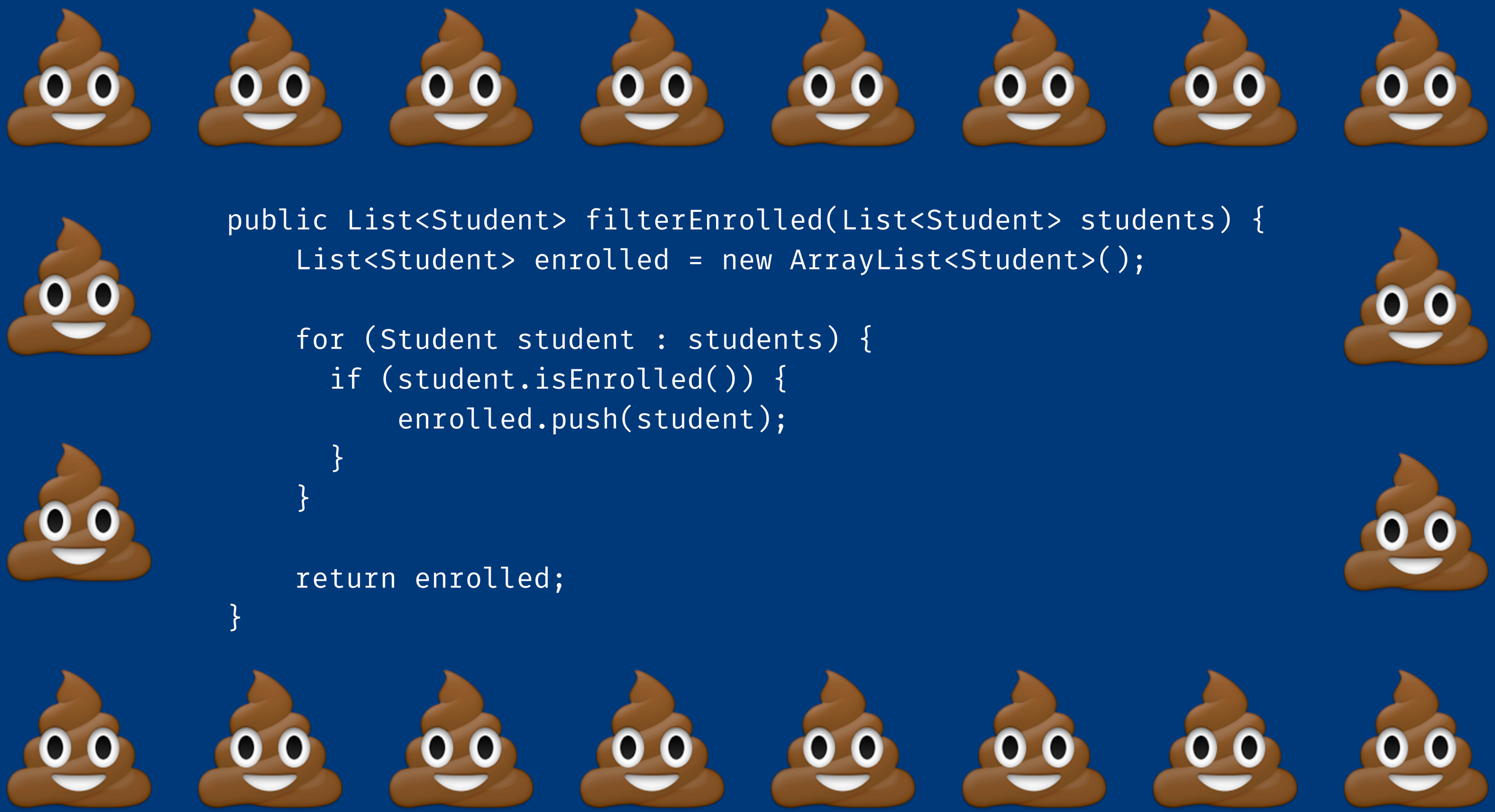
Velocidade de desenvolvimento
Práticas ágeis (XP nasceu no Smalltalk)
Poder de síntese (mais com menos)
Liberdade na escolha de ferramentas
REPL
Duck typing
Prototipagem mais barata

Performance inferior
Menos ferramentas de análise estática
Detecção tardia de erros (em runtime)
Menos chances para otimização

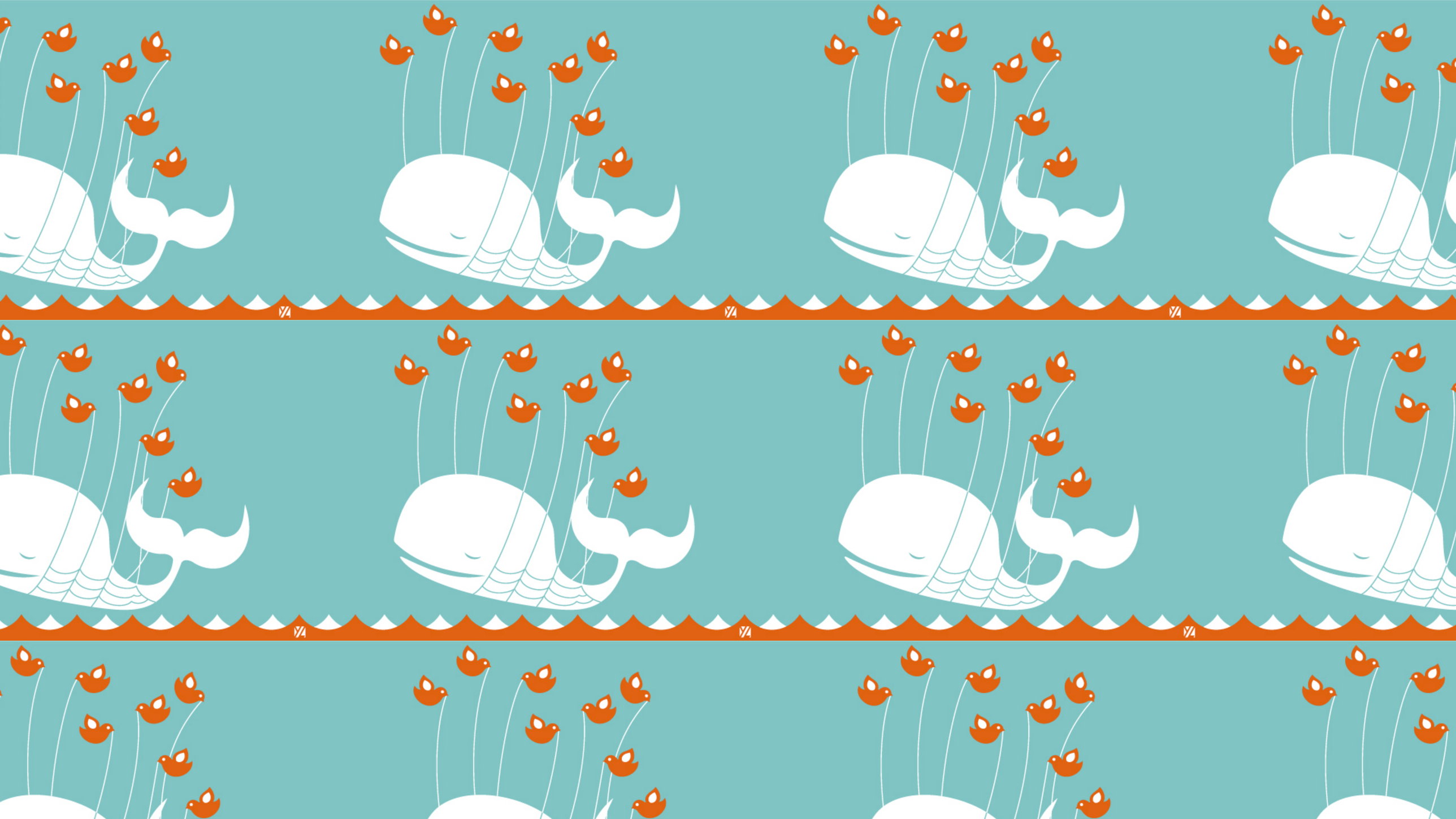
No prato estático

Erros capturados ao compilar
Performance superior
Ferramentas avançadas de análise
Ferramentas avançadas de reestruturação
Otimizações automáticas

Excesso de cerimônia
Verbosidade
Ciclos lentos de iteração
Rigidez ao modificar por conta dos tipos


public List<Student> filterEnrolled(List<Student> students) {
List<Student> enrolled = new ArrayList<Student>();
for (Student student : students) {
if (student.isEnrolled()) {
enrolled.push(student);
}
}
return enrolled;
}


enrolled = students.filter(&:enrolled?)



A resposta, como sempre, é olhar para o que a Academia estava fazendo 30+ anos atrás

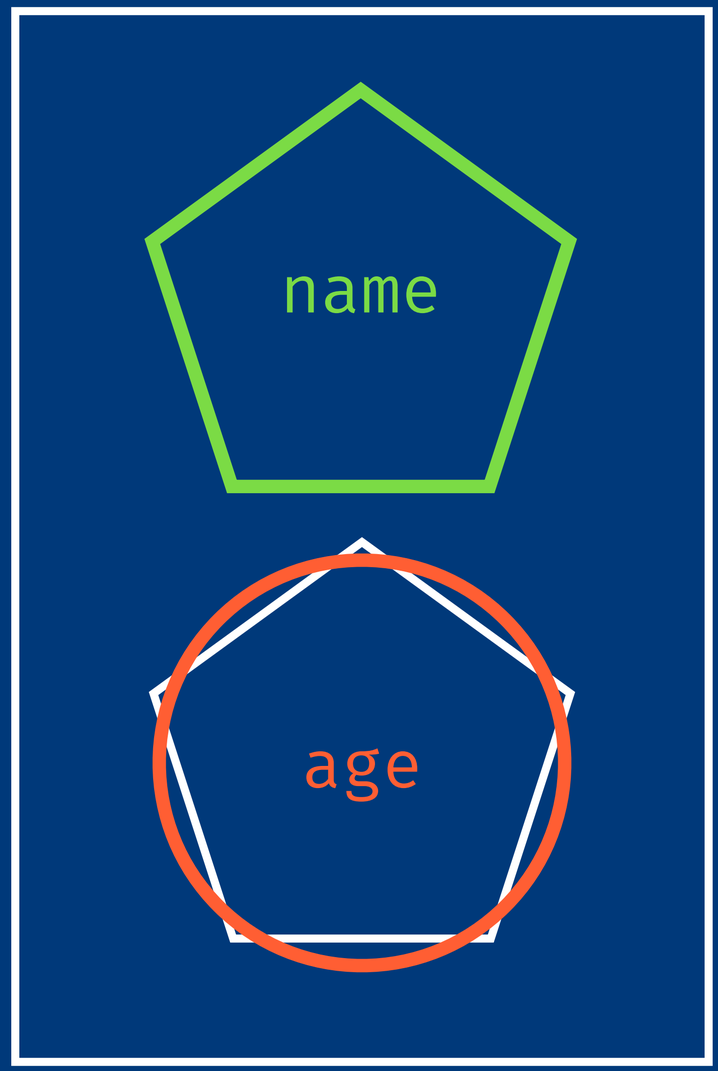
Inferência de tipos de Hindley-Milner

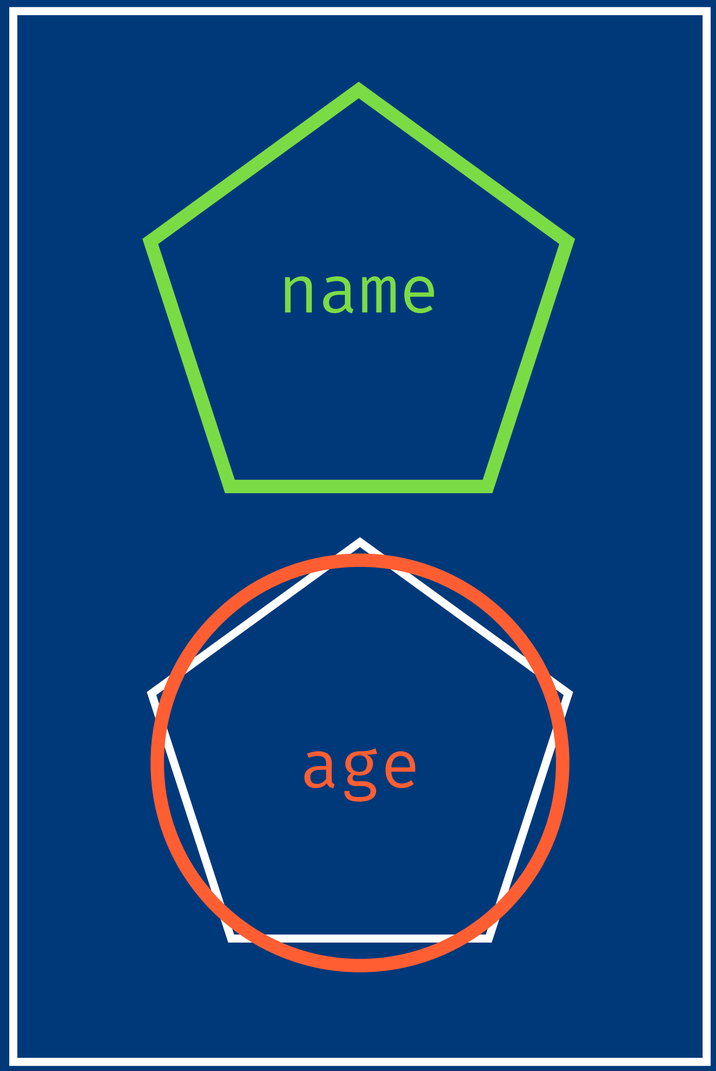
```
public void printInfo(String name, String age) {
    System.out.println(“My name is ”  + name +
        “ and I’m ” + age + “years old.”);
}

String name = “The Doctor”;
int age = 903;
printInfo(name, age);
```

```
public void printInfo(String name, String age) {
    System.out.println(“My name is ”  + name +
        “ and I’m ” + age + “years old.”);
}

String name = “The Doctor”;
int age = 903;
printInfo(name, age);
```

int cannot be converted to String


name
age


name
age

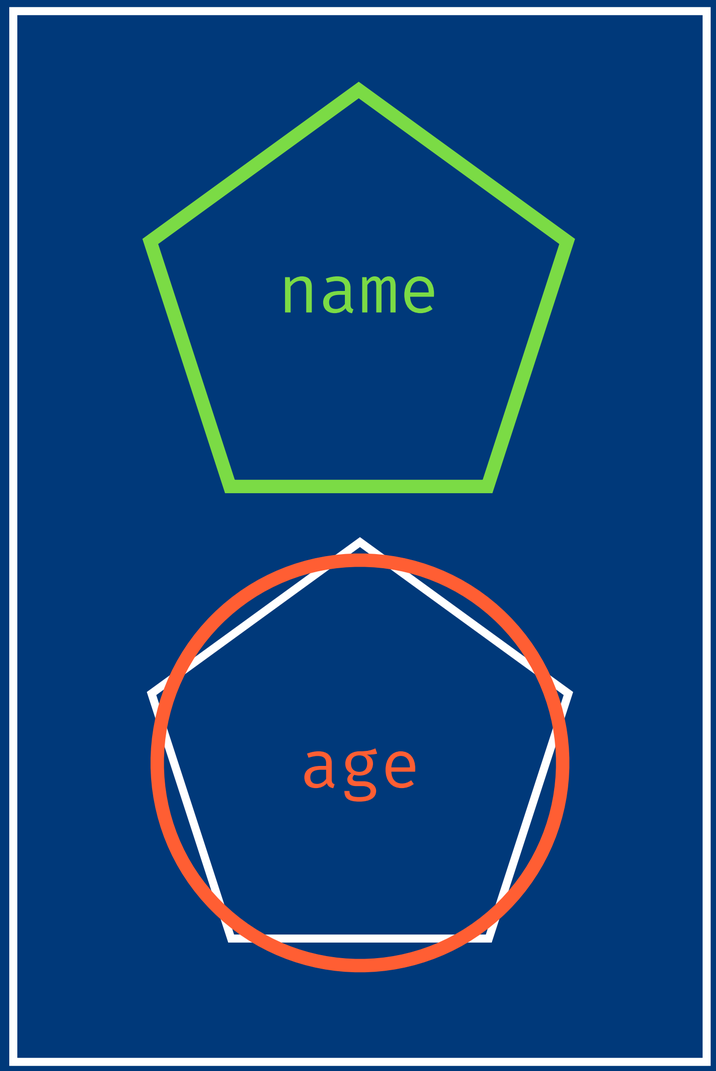

name
age

```
printInfo name age = do
  putStrLn $ “My name is ” ++ name ++ “and I’m ” ++ age ++
“ years old”

main = do
  printInfo name age

  where
    name = “The Doctor”
    age = 903
```


name
age


name
age

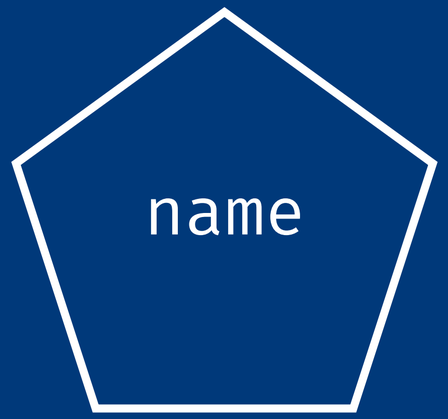

name
age


name
age

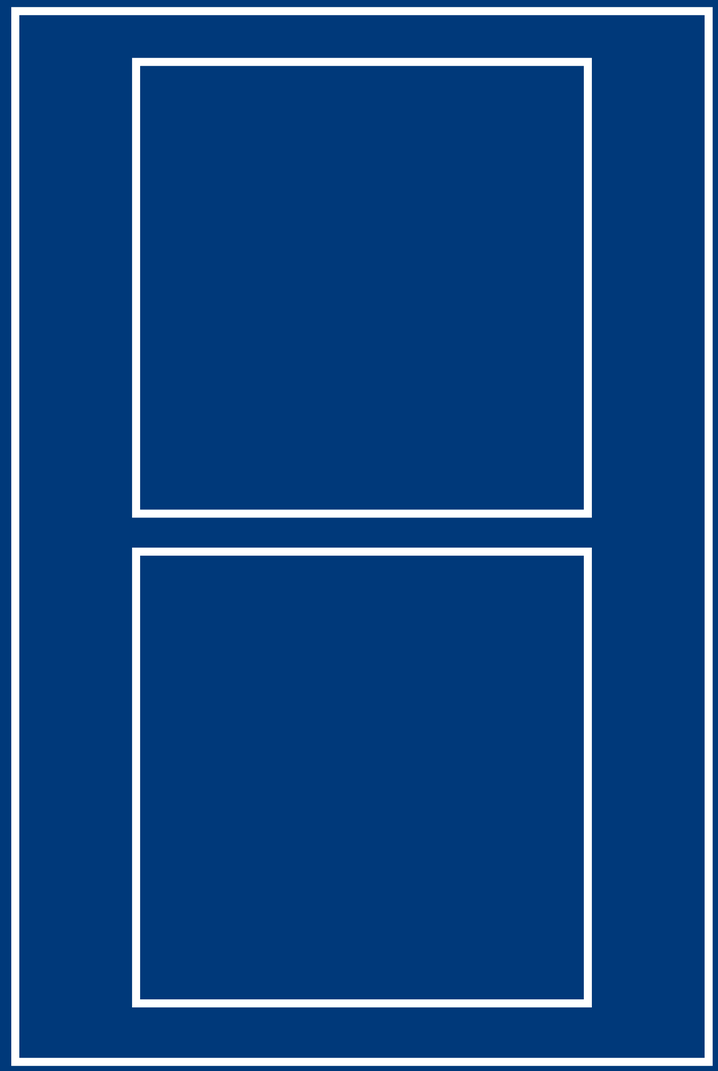
```
$ ghci printInfo.hs
*Main> :t printInfo
printInfo :: [Char] → [Char] → IO ()
*Main> :t putStrLn
putStrLn :: String → IO ()
*Main> :i String
type String = [Char]
```

```
printInfo name age = do
  putStrLn $ “My name is ” ++ (show name) ++ “and I’m ” ++
(show age) ++ “ years old”

main = do
  printInfo name age

  where
    name = “The Doctor”
    age = 903
```

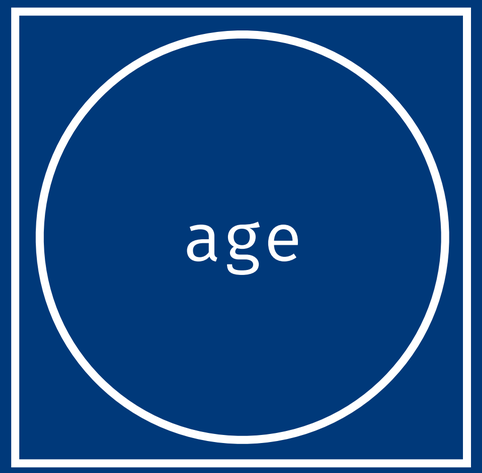
```
$ ghci printInfo.hs
*Main> :t printInfo
printInfo :: (Show a, Show a1) ⇒ a → a1 →
IO ()
```

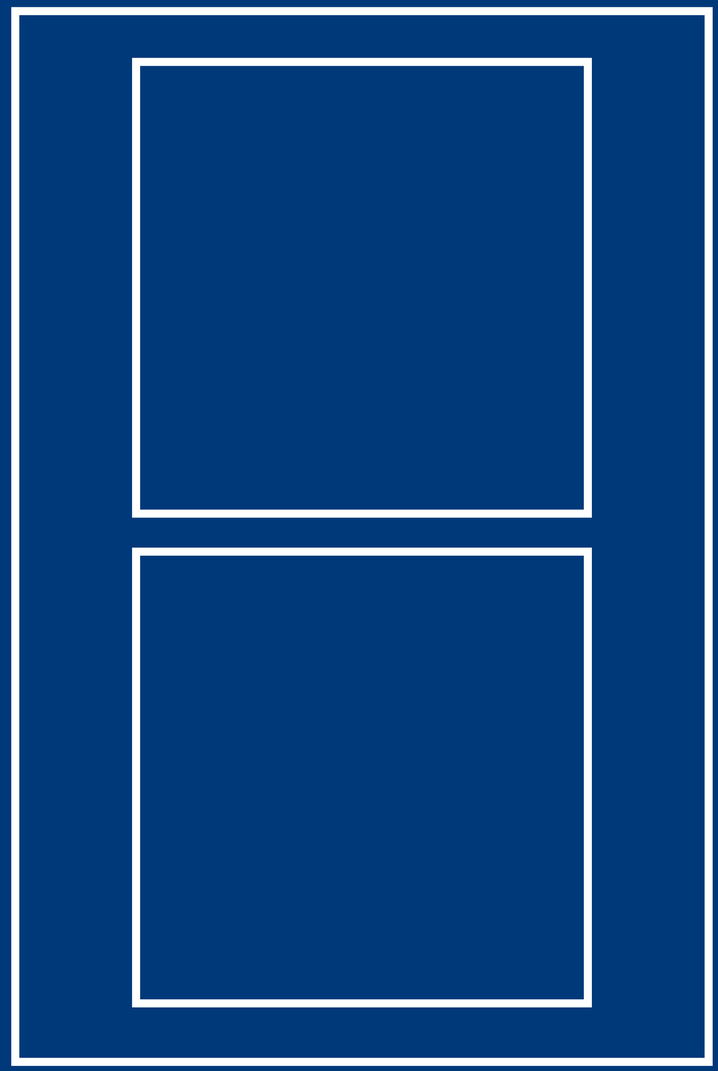

name

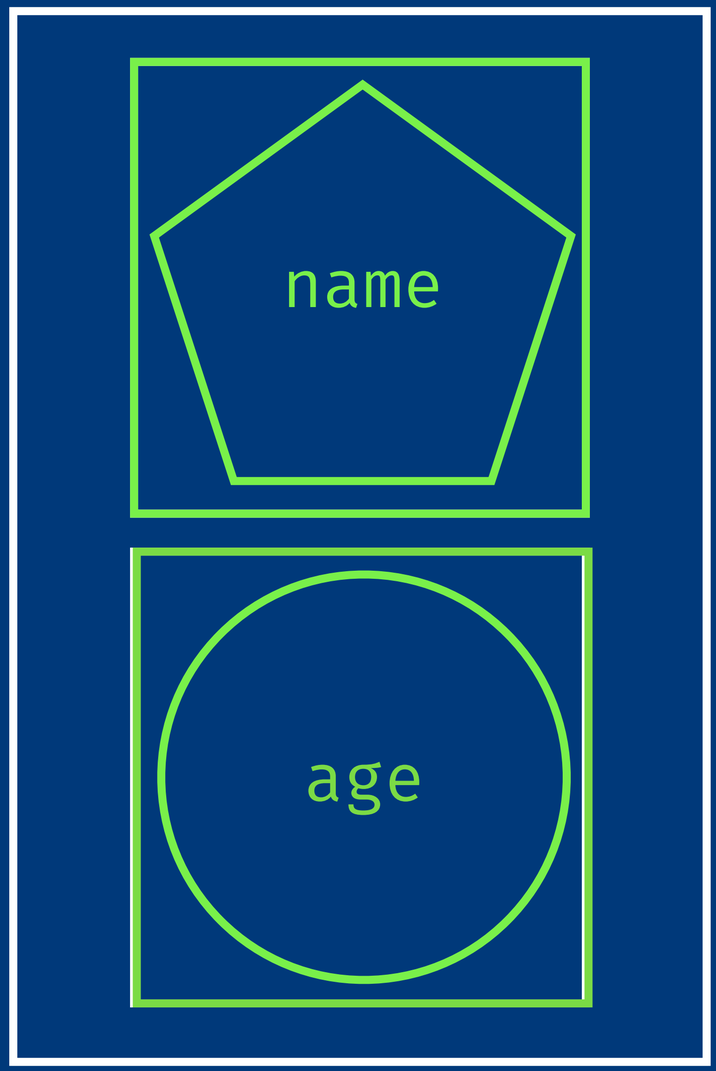

age




name


age




name
age

Sistemas de Efeitos

```haskell
main :: IO ()
main = do
  user ← fetchFromDatabase(10)
  sendNudes user
```

Haskell

```elm
fetchProfile : User → (Effects Action)
fetchProfile user =
    Http.get profileDecoder
user.profileUrl
    |> Task.toMaybe
    |> Task.map UpdateProfile
    |> Effects.task
```

Elm

Paralelismo gratuito

```haskell
updateAnalytics :: IO (AverageResponseTime)
updateAnalytics = do
  dataA ← dataFromSystemA — IO (AverageResponseTime)
  dataB ← dataFromSystemB — IO (AverageResponseTime)
  dataC ← dataFromSystemC — IO (AverageResponseTime)

  return $ AverageResponseTime (dataA + dataB + dataC) / 3
```

Liberação determinística de recursos

```
struct Foo<'a> {
    x: &'a i32,
}
```

Rust

"Estou cagando e andando pra Haskell, Elm, Idris, essas bagaças todas. O mundo real é JavaScript!

-Vocês e Todo Mundo


flow

```
function mul(x) {
  return x * 10;
}

mul(“foo"); // ⇒ NaN
```

```
function mul(x) {
  return x * 10;
}

mul(“foo"); // ⇒ NaN
```

```
/* @flow */
function mul(x) {
  return x * 10;
}

mul(“foo"); // ⇒ NaN
```

```
6: mul("foo");
    ^^^^^^^^^^ function call
 3:   return x * 10;
             ^ string. This type is incompatible with
 3:   return x * 10;
             ^^^^^^ number
```

```
/* @flow */

function fullName(person) {
  return person.firstName + " " + person.lastName;
}

console.log(fullName(null));
```

```
8: console.log(fullName(null));
                  ^^^^^^^^^^^^^^^ function call
  4:   return person.firstName + " " + person.lastName;
                                                ^^^^^^^^

property `lastName`. Property cannot be accessed on possibly
null value
  4:   return person.firstName + " " + person.lastName;
                                       ^^^^^^ null
```

```
/* @flow */

function fullName(person) {
  if (person == null) {
    return "John Doe";
  } else {
    return person.firstName + " " + person.lastName;
  }
}

console.log(fullName(null));
```

```
/* @flow */

function foo(x) {
  return x.length;
}

var res = foo("Hello") + foo(42);
```

Dinâmico

Estático

Dinâmico

Estático

Estático || Dinâmico

Forte > Fraco

Implícito > Explícito