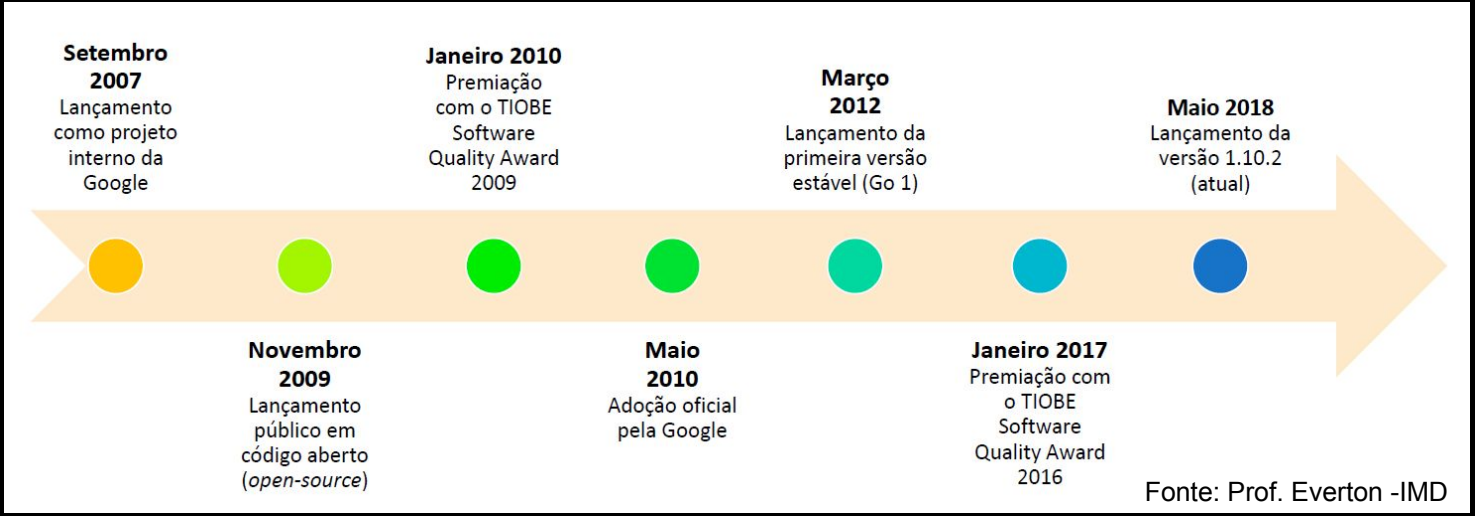
Go lang

Go e suas especificidades

Adelson Lira
Paulo Costa

A Linguagem da Google

O Google estava com um problema, pois muitos dos seus sistemas eram feitos em C++ e em C, e o processo de compilar esses programas, para gerar um executável, era complicado e demorado. Com isso, os engenheiros do Google tiveram a ideia de criar uma nova linguagem de programação, surgindo daí o **Go**.



Fonte: Prof. Everton -IMD

A Linguagem da Google

- Go possui tipagem forte e estática. Porém introduz uma forma curta de declaração de variáveis baseada em inferência de tipos, evitando redundâncias e produzindo um código mais sucinto.
- Clássico x Moderno (C, C++, Java, JavaScript)
- Duck typing: se faz “quack” como um pato e anda como um pato, então provavelmente é um pato.
- Suporta o uso de ponteiros.

Vantagens da Go

- Compiled & Rapid
- Concurrent & Fast Running
- Readable & Maintained
- Open-sourced & Accessible

Vantagens da Go

Go combina a leveza, facilidade de uso e expressividade do Javascript e Python com a eficiência e segurança de linguagens estaticamente tipadas tradicionais como C/C++ e Java

Go possui Suporte nativo a concorrência.

Vantagens da Go x Java

source	secs	mem	gz	cpu	cpu load
Go	**2.04**	8,976	603	2.04	1% 0% 100% 0%
Java	3.14	37,080	938	3.35	4% 4% 1% 98%

The Computer Language Benchmarks Game

Vantagens da Go x Javascript

source	secs	mem	gz	cpu	cpu load
Go	**2.04**	8,976	603	2.04	1% 0% 100% 0%
Node js	14.47	62,316	530	14.53	13% 1% 1% 87%

The Computer Language Benchmarks Game

Vantagens da Go x Python

source	secs	mem	gz	cpu	cpu load
Go	**5.47**	31,280	905	21.73	99% 99% 99% 100%
Python 3	279.68	49,344	688	1,117.29	100% 100% 100% 100%

The Computer Language
Benchmarks Game

Estrutura

Um código fonte em Go consiste basicamente de três partes:

- Declaração de pacote (package);
- Importação de pacotes (imports);
- Declarações de tipos, variáveis e funções;

Estrutura

```go
package main

import "fmt"

func main() {
    fmt.Println("Hello World!")
}
```

Tipos e variáveis

- boll, string, int (8, 16, 32, 64), uint (8, 16, 32, 64), float (32, 64), complex (64, 128);
- Declaração de variáveis:
  - Var nome string = “José”
  - nome := “José"

Tipos e variáveis

Funções em go lembram o Javascript:

```
func soma(a int, b int) int{
return a + b
}
```

Arrays e Slice

Arrays em Go são estáticos e não podem sofrer alterações em seu tamanho. Para lidar com arrays dinâmicos, Go provê slices

- Um slice é uma referência para um segmento contíguo de um array subjacente
- O tamanho de um slice pode mudar dinamicamente durante a execução do programa
- Slice cria um novo array com o dobro do tamanho do array anterior sempre que o array estoura em sua capacidade.

Arrays e Slice

Arrays em Go são estáticos e não podem sofrer alterações em seu tamanho. Para lidar com arrays dinâmicos, Go provê slices

- Um slice é uma referência para um segmento contíguo de um array subjacente
- O tamanho de um slice pode mudar dinamicamente durante a execução do programa
- Slice cria um novo array com o dobro do tamanho do array anterior sempre que o array estoura em sua capacidade.

Fluxo de controle

Em sua maioria, similares às existentes nas linguagens C/C++ e Java

- Condição: if/else
- Seleção: Switch
- Iteração: For

Diferenças

- O uso de parênteses não é obrigatório
- O uso de chaves é obrigatório
- Não existe While ou Do While para iteração

Go é Orientada a Objetos?

Classes

Go não faz uso de classes. Classes são substituídas por Structs, que são coleções tipadas de campos. Ex:

```
package main

import "fmt"

type person struct {
    name string
    age  int
}

func main() {
```

Herança

Não há o uso de herança em Go. Ao invés de herança, pode-se usar composição

```
type Dog struct {
    Animal
}
type Animal struct {
    Age int
}
```

Interfaces

```go
package main

import "fmt"

type Figura interface {
    area() int
}

type Retangulo struct {
    largura, altura int
}

type Quadrado struct {
    lado int
}

func (r Retangulo) area() int {
    return r.largura * r.altura
}

func (q Quadrado) area() int {
    return q.lado * q.lado
}

func main() {
    ret := Retangulo{3, 4}
    quad := Quadrado{2}
    fmt.Println("Area do retangulo:", ret.area())
    fmt.Println("Area do quadrado:", quad.area())
}
```

Os tipos `Retangulo` e `Quadrado` implementam a interface `Figura` por definirem cada um o método `area`

Encapsulamento

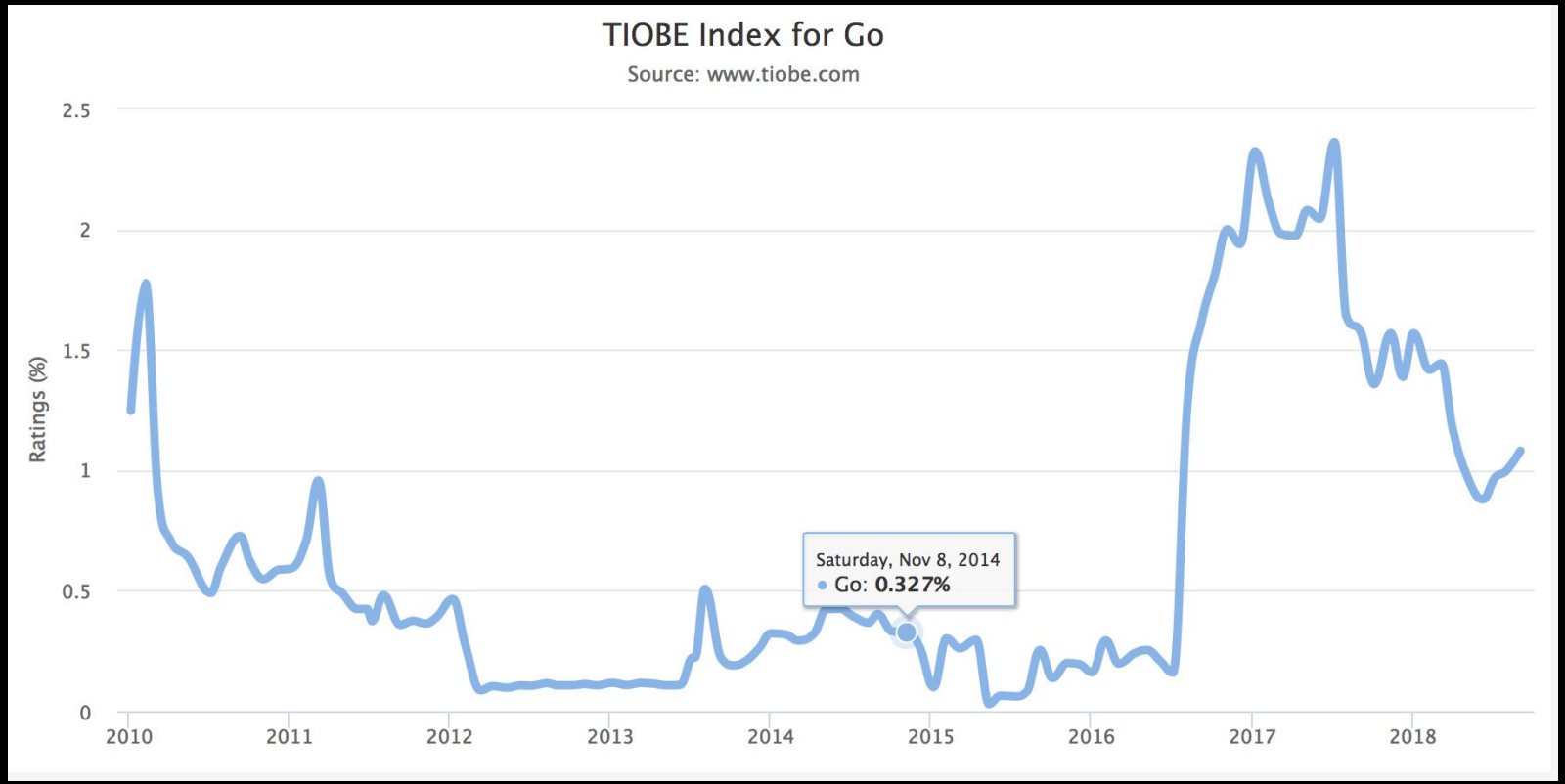
Go encapsula a nível do package. Nomes que começam com letras minúsculas estão visíveis apenas para o package. Letras maiúsculas são públicas

Go é Orientada a Objetos?

Go é Orientada a Objetos?

Sim e não

Go no Go


TIOBE Index for Go
Source: www.tiobe.com
Ratings (%)
2.5
2
1.5
1
0.5
0
2010
2011
2012
2013
2014
2015
2016
2017
2018
Saturday, Nov 8, 2014
Go: 0.327%

Go no Go

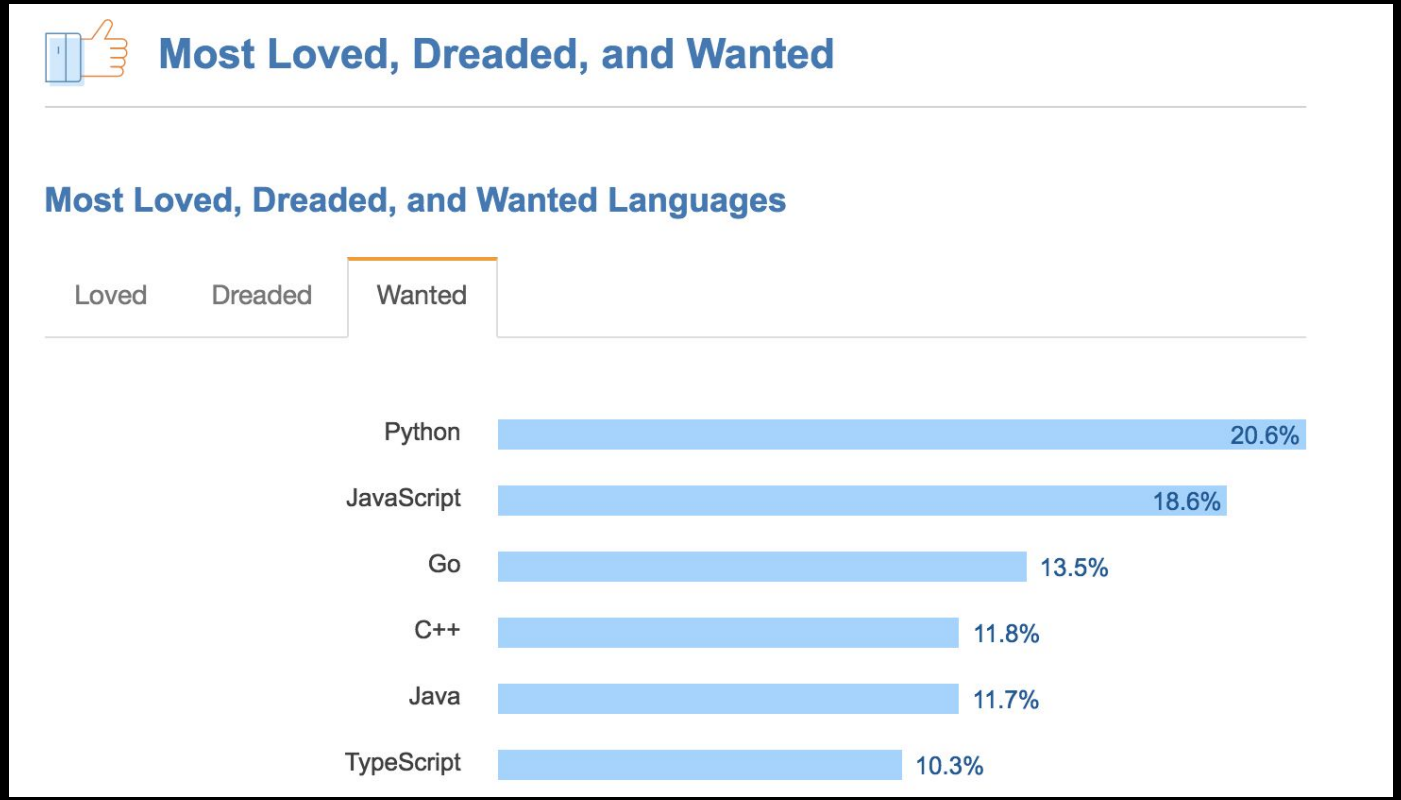
Most Loved, Dreaded, and Wanted

Most Loved, Dreaded, and Wanted Languages

Loved | Dreaded | Wanted

Rust	73.1%
Smalltalk	67.0%
TypeScript	64.1%
Swift	63.9%
Go	63.3%
Python	62.7%

Go no Go


Most Loved, Dreaded, and Wanted
Most Loved, Dreaded, and Wanted Languages
Loved
Dreaded
Wanted
Python
20.6%
JavaScript
18.6%
Go
13.5%
C++
11.8%
Java
11.7%
TypeScript
10.3%

Para saber mais

https://golang.org/