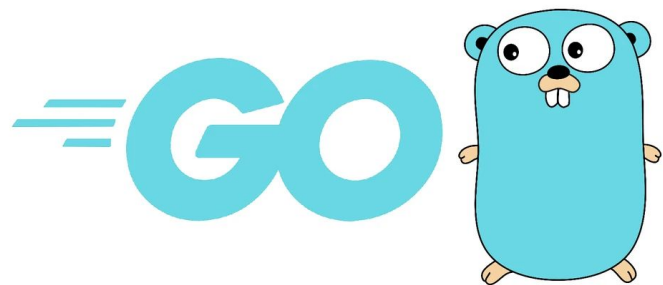


# Lenguajes especiales para conurrencia

Técnicas de Programación Concurrente



# Origen y Características

- Iniciado en 2007 en Google. Primera versión estable en 2012. (Rust 2015)
- Compilado a código de máquina
- Fuertemente tipado con inferencia de tipos
- Structural typing ("duck typing" pero chequeado estáticamente)
- Garbage collector
- Gorutines y channels
- Punteros sin aritmetica. Slices
- *Do not communicate by sharing memory; instead, share memory by communicating.*
- No tiene manejo de errores

# Goroutines

- Son green threads administradas por el runtime de go
- Por defecto tienen un stack de [2kb](#), que puede [moverse para crecer](#)
- El ambiente de ejecución esta incluido en el lenguaje
- Cualquier función puede ejecutarse de forma concurrente con la palabra reservada **go**. main misma corre como goroutine
- Le ejecución se planifica sobre un pool de **GOMAXPROCS** hilos
- Si bien su nombre parte de 'coroutine' no requiere ser explícitamente colaborativa

# Ejemplos

- Hello channels world
- Consumidor infinito
- Problema del banquero
- Múltiples consumidores
- Buffered
- Select

# Estadísticas - JetBrains

One out of every two developers is planning to adopt a new language. The top choices for next languages are **Go, Rust, Kotlin, TypeScript, and Python.**

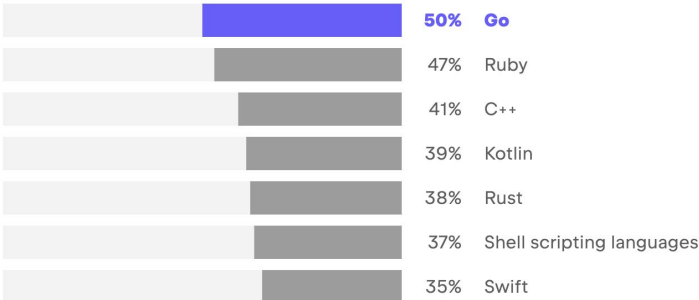
The most favorite programming languages are **Kotlin, C#, Python, Rust, and Java.**

## Which technologies do you find promising?

Based on answers to a free-response question.

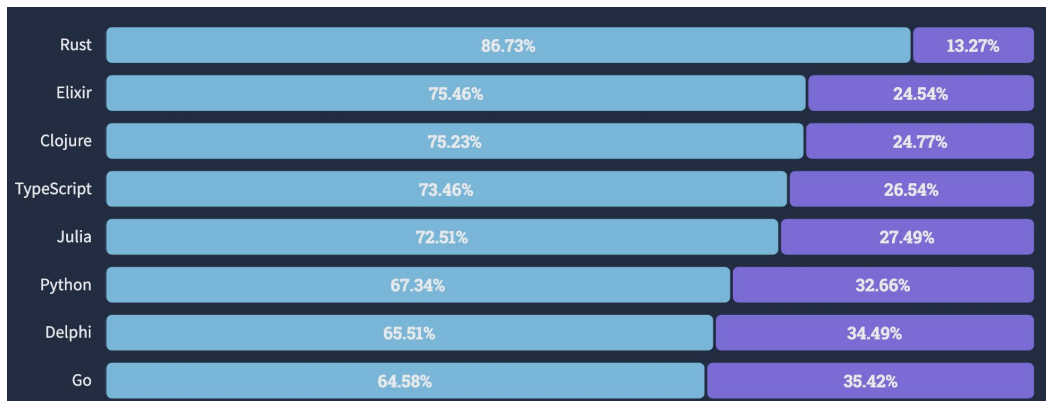


## Share of top-paid employees by primary language

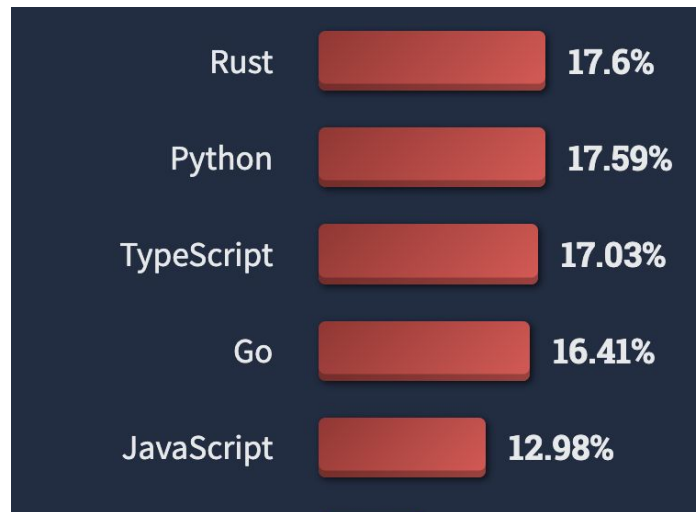


# Estadísticas - Stackoverflow

## Loved



## Wanted





elixir



# Origen y Características

- Creado en 2011 por Jose Valim, desarrollador de Ruby on Rails.
- Corre sobre la máquina virtual de Erlang (BEAM VM)
- Procesos como elemento de diseño que se comunican via mensajes. Actores
- Tolerancia a fallas - *Let it crash*
- Funcional puro
- DSLs
- Lo usa Discord. Lo usa Whatsapp (Erlang)
- Los procesos se ejecutan sobre scheduler propio con un pool de runners igual a la cantidad de CPUs

# OTP (Open Telecom Platform)

- Biblioteca con funcionalidades varias
- Protocolo para definir de forma estandarizada un proceso ("actor") via extend GenServer
- Supervisores para monitoreo y reemplazo de procesos

# Ejemplos

- Hello world
- Pattern matching
- OTP (Open Telecom Platform)
- Supervisores

# Estadísticas - Most paying





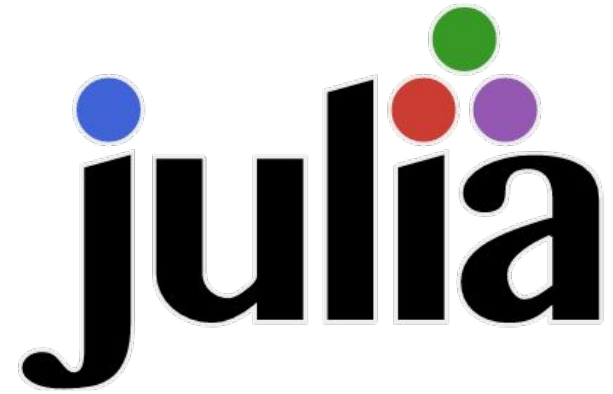
Clojure

# Origen y Características

- Iniciado en 2005, primer release en 2007
- Dialecto de Lisp (code is data!)
- Corre sobre la JVM
- Software Transactional Memory
- Atlassian usa Clojure para implementar el backend de su realtime collaboration

# Ejemplos

- Hello World
- STM
- swap
- agent

The logo for the Julia programming language, featuring the word "julia" in a bold, lowercase, black sans-serif font. The letter 'j' has a blue dot above it. The letter 'i' has a red dot above it. The letter 'l' has a green dot above it. The letter 'a' has a purple dot above it.

**julia**



# Origen y Características

- Iniciado en 2009, primer release en 2012
- Pensado para scientific computing, machine learning, data mining, large-scale linear algebra, distributed and parallel computing
- Lightweight "green" threading (coroutines)
- Unobtrusive yet powerful type system
- Call C functions directly (no wrappers or special APIs needed)
- SIMD support out of the box
- Python-like syntax

# Ejemplos

- Hello world
- @Threads
- Distributed

# Referencias

## GO

A Tour of Go <https://tour.golang.org/welcome/1>

Effective Go [https://golang.org/doc/effective\\_go](https://golang.org/doc/effective_go)

The Go Memory Model - <https://golang.org/ref/mem>

Go Concurrency Patterns <https://talks.golang.org/2012/concurrency.slide#1>

Advanced Go Concurrency Patterns <https://talks.golang.org/2013/advconc.slide#1>

Contiguous stacks <https://docs.google.com/document/d/1wAaf1rYoM4S4gtnPh0zOIGzWtrZfQ5suE8qr2sD8uWQ/pub>

## Elixir

<https://elixir-lang.org/>

<https://hexdocs.pm/elixir/1.12/Supervisor.html>

## Julia

<https://julialang.org/blog/2012/02/why-we-created-julia/>