Descomplicando linguagens

Novas linguagens no futuro de empresas

Eugenio Cunha

26 de agosto de 2019

Matermaq Software

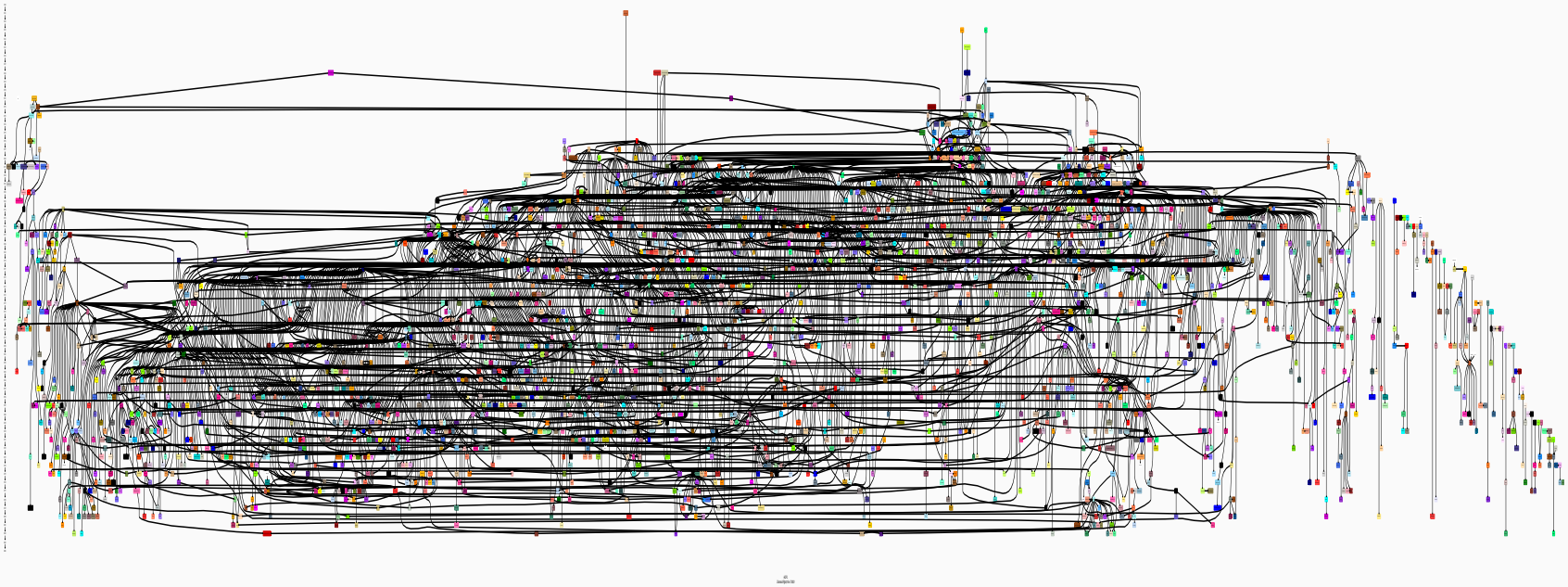
Conteúdo

Introdução

Unicórnio das linguagens

Encontrar a linguagem de programação perfeita?

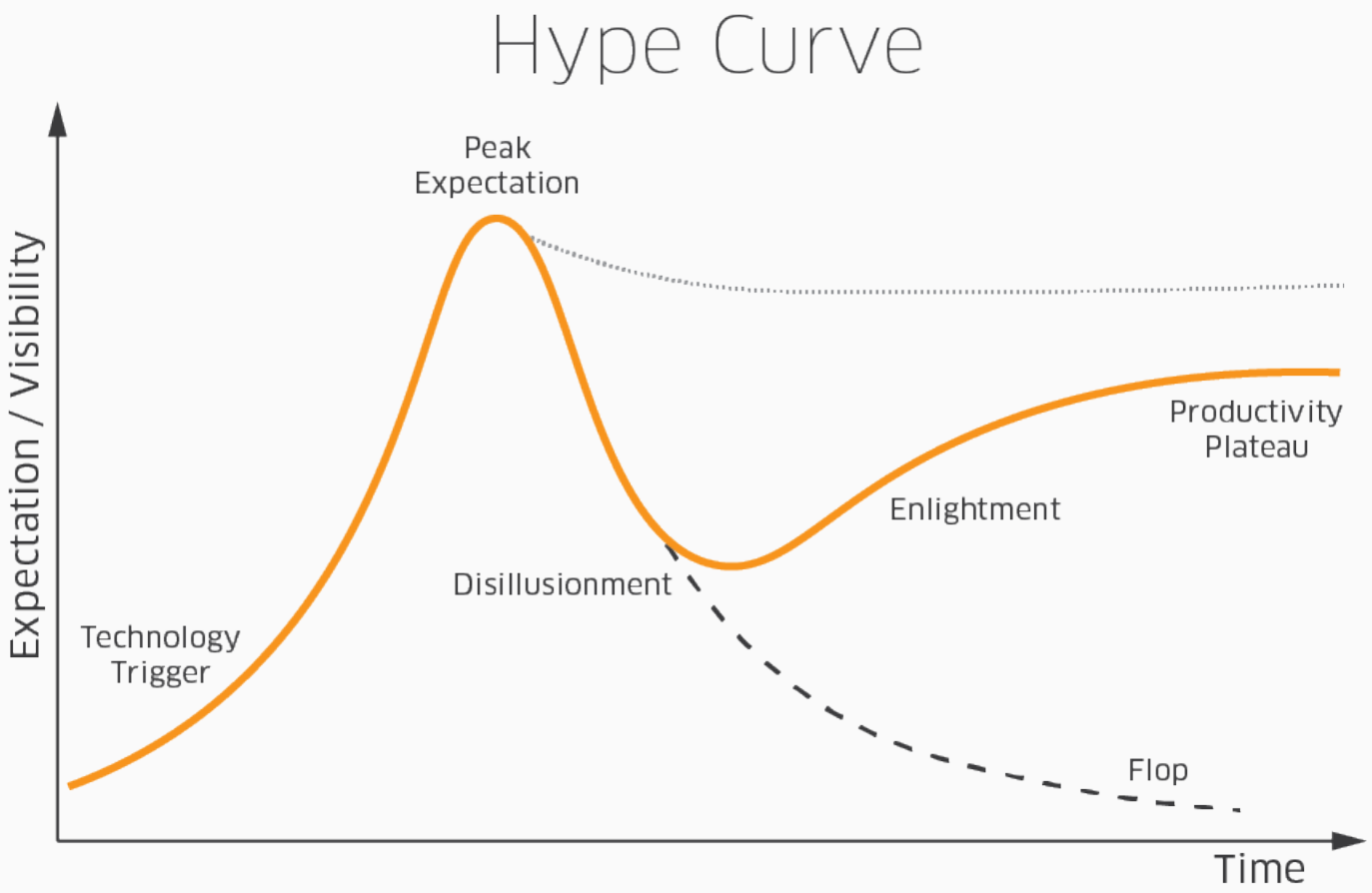
HOPL (A history of the history of programming languages)



Desde da década de 40 mais de 8.945 linguagens de programação foram criadas.

Tendência do Mercado

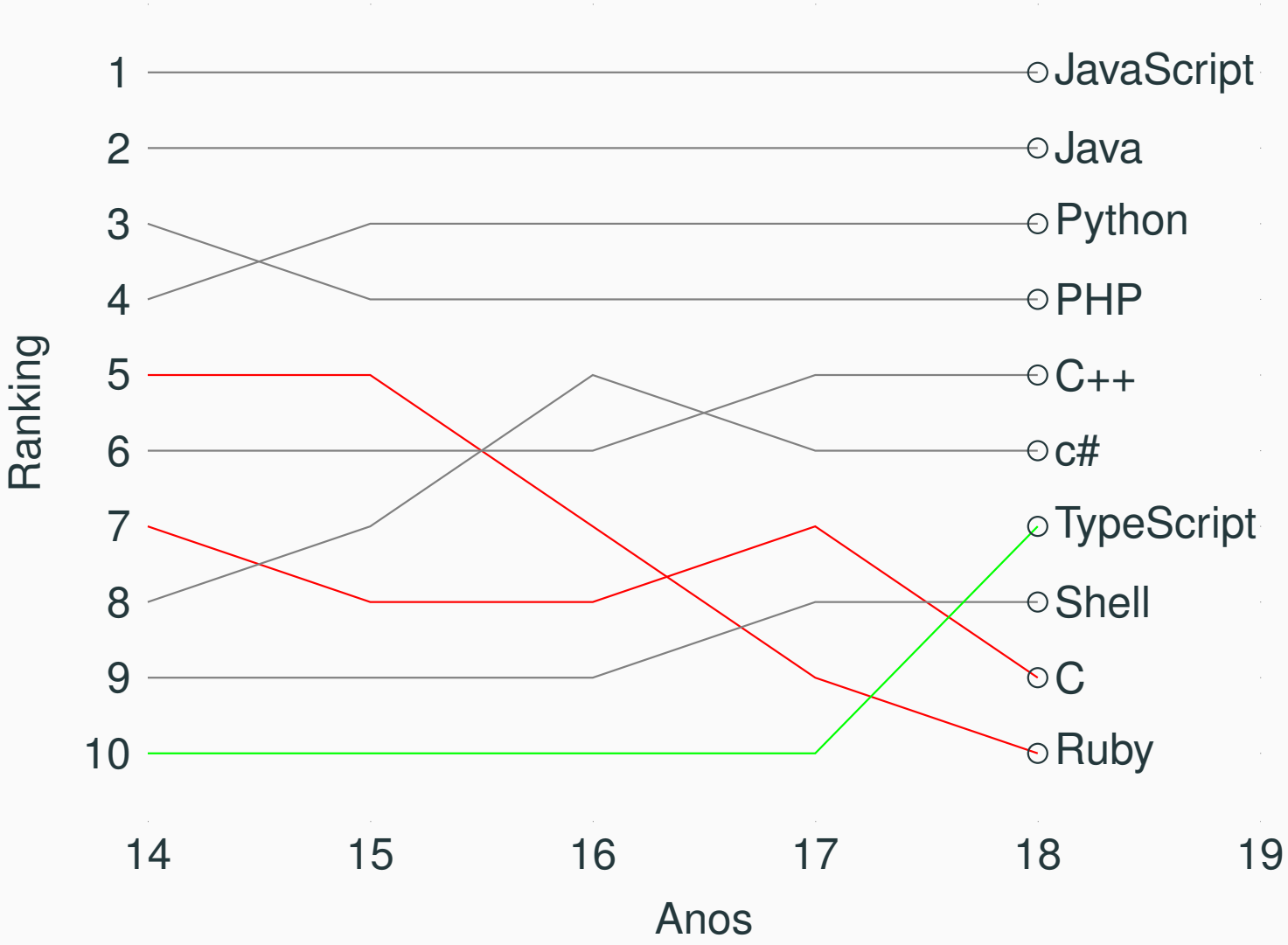
Hype Curve Technology



Onde encontrar essas tendências?

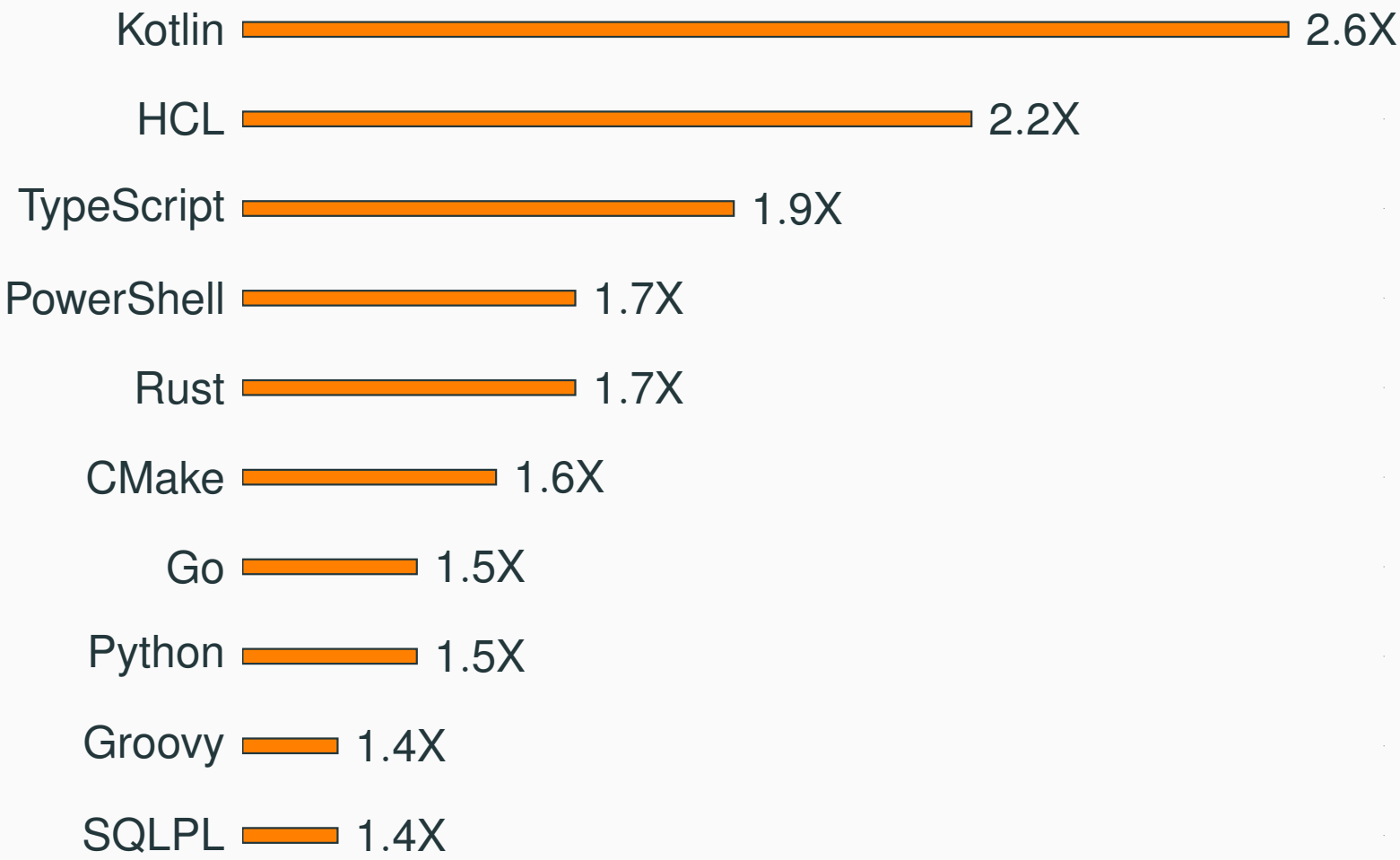
GitHub tendências e insights https://octoverse.github.com/

Linguagens de 2018



Principais linguagens ao longo do tempo

Linguagens com crescimento mais rápido

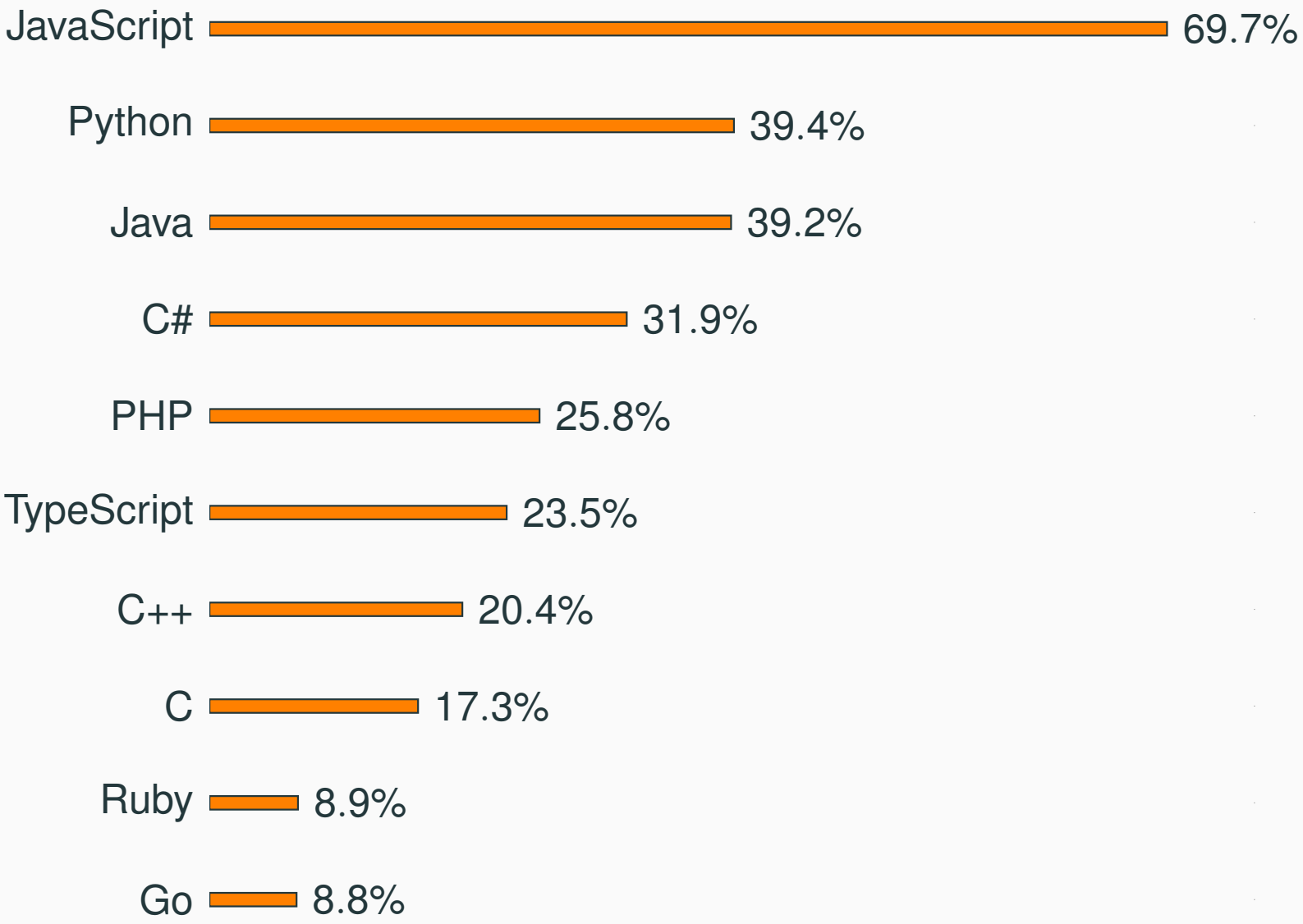


Tendências em direção a mais linguagens com tipos estáticos e interoperabilidade

Stack Overflow

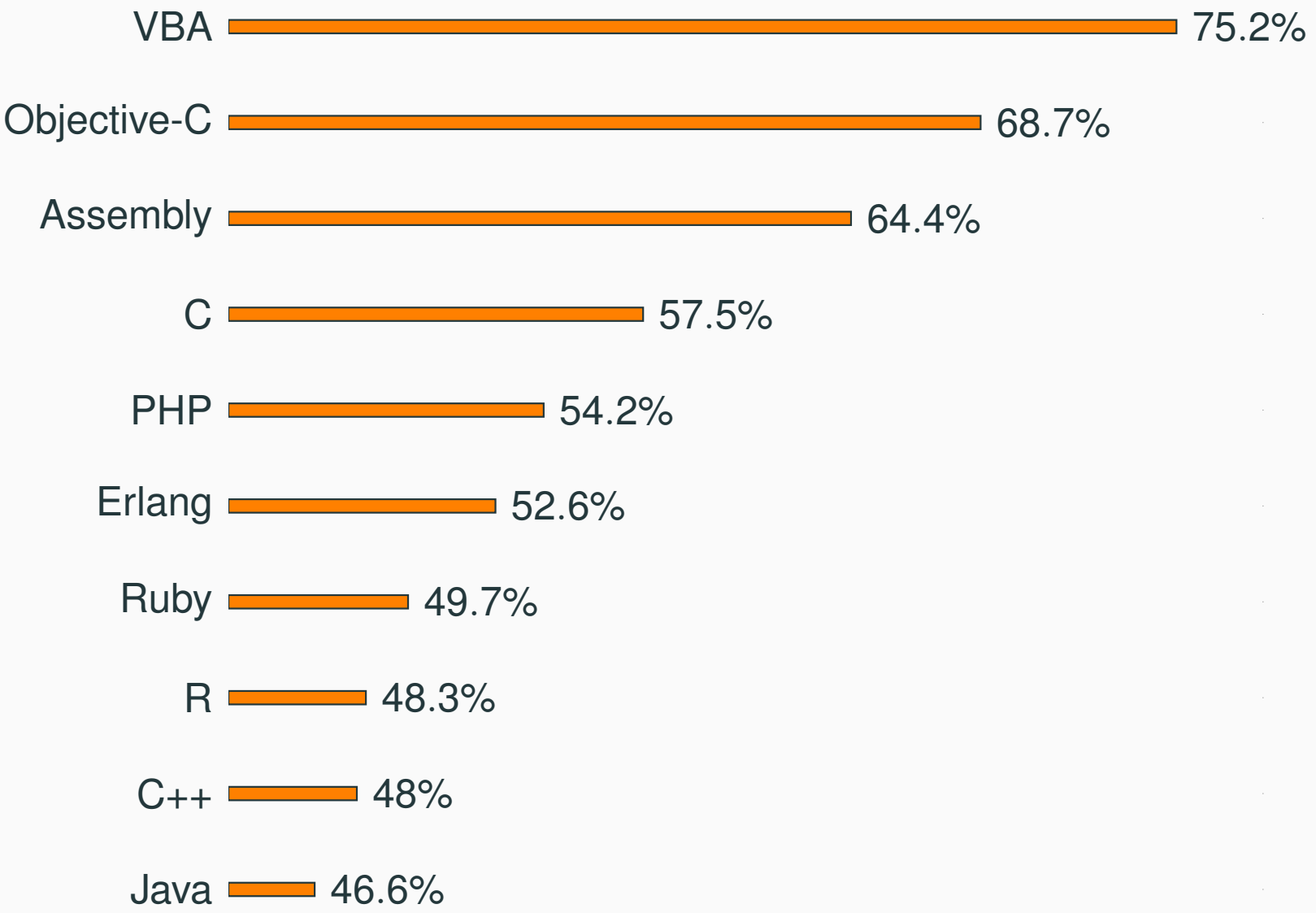
Quase 90.000 mil desenvolvedores profissionais participaram
https://insights.stackoverflow.com/survey/2019

Linguagens mais populares



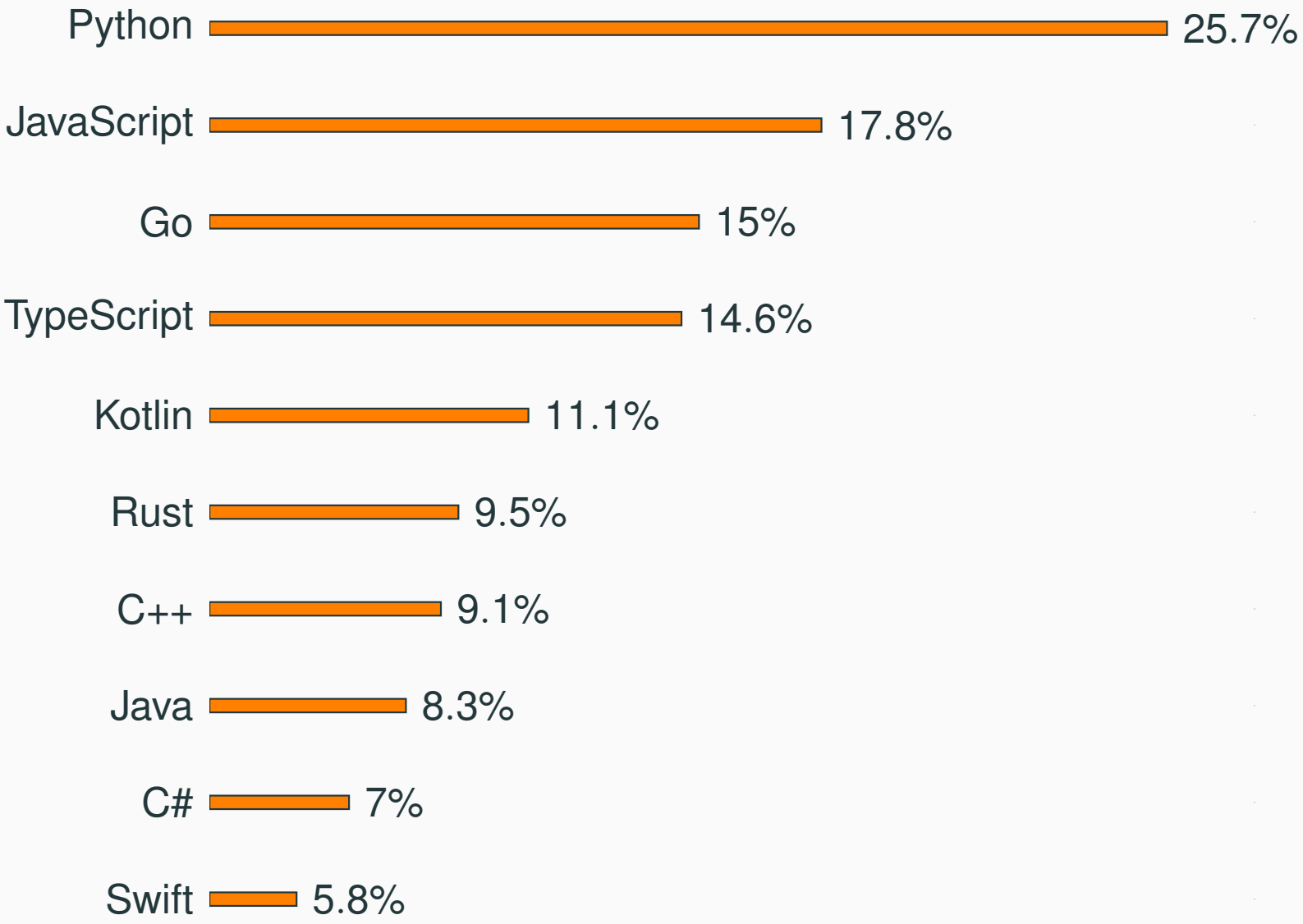
% de desenvolvedores profissionais

Linguagens mais odiadas



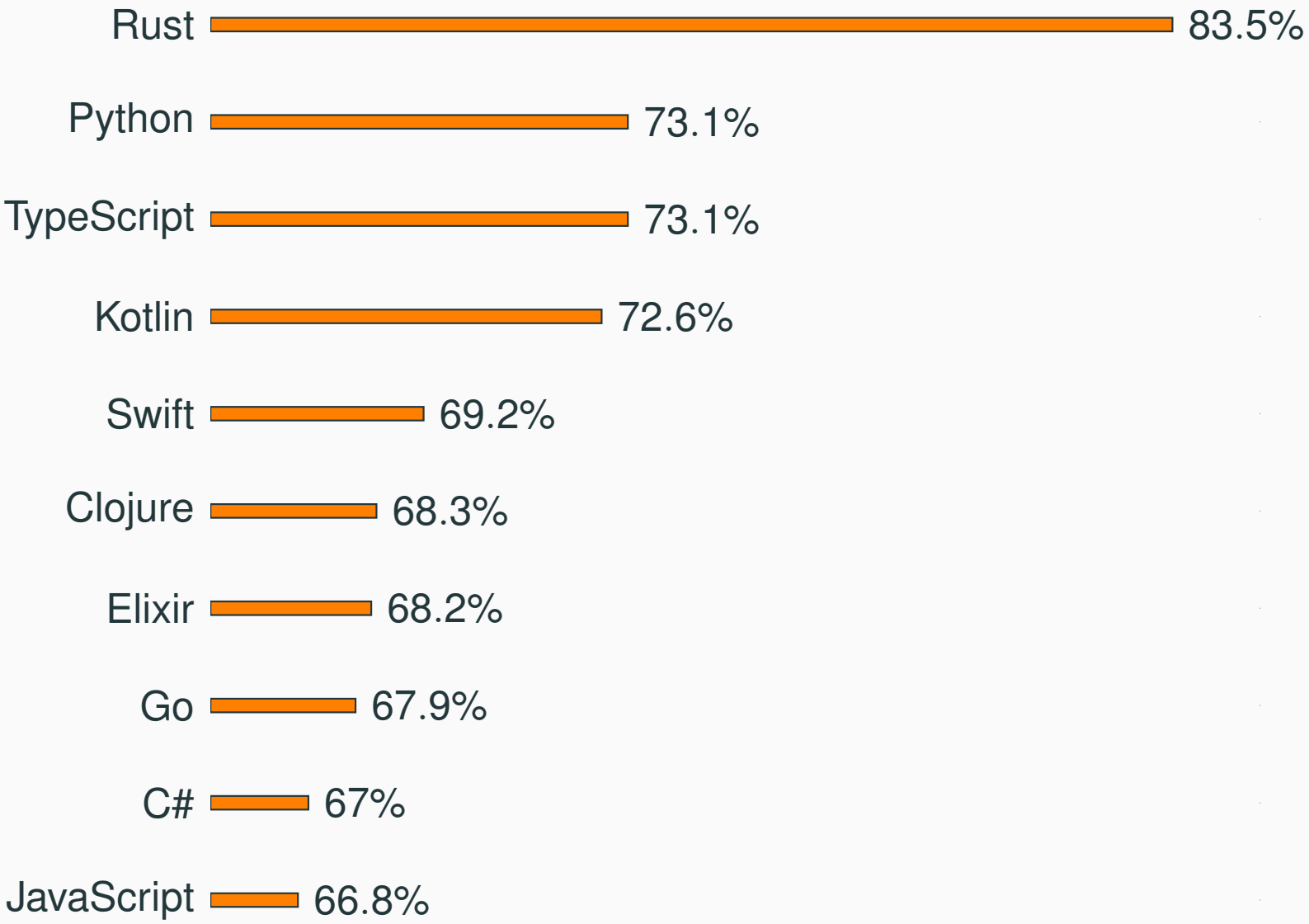
% de desenvolvedores que não demonstraram interesse de usar

Linguagens mais desejadas



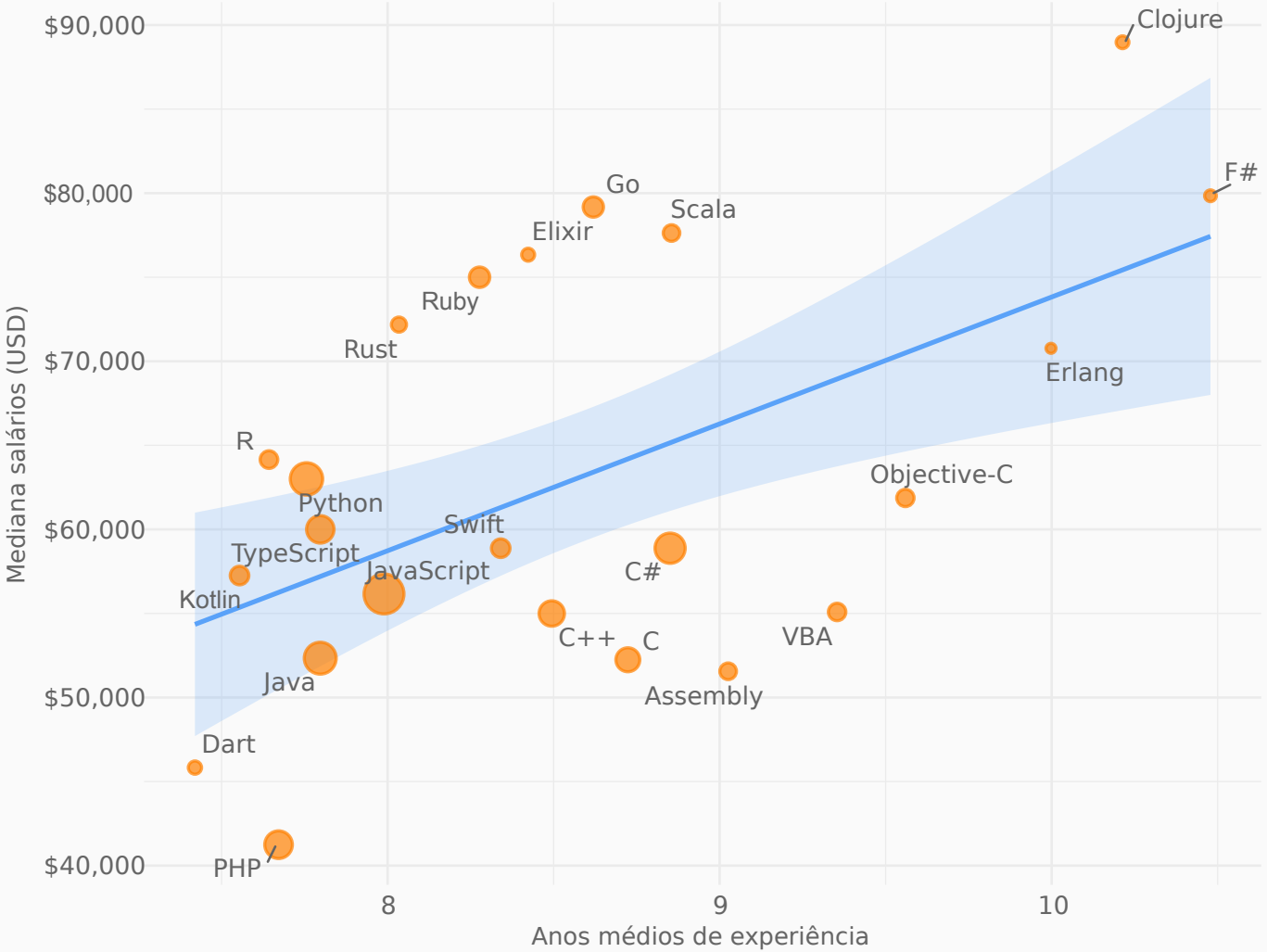
% de desenvolvedores que demonstraram interesse de aprender

Linguagens mais amadas



% de desenvolvedores que demonstraram interesse de continuar

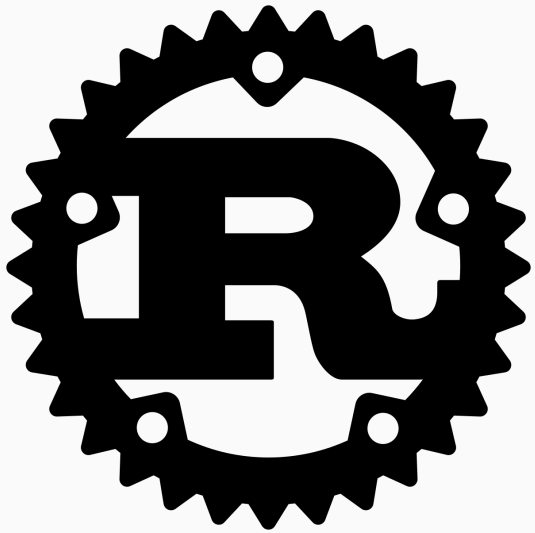
Salário e experiência por linguagens



Salário | Experiência | Quantidade

Nova Linguagem (Rust)

Rust

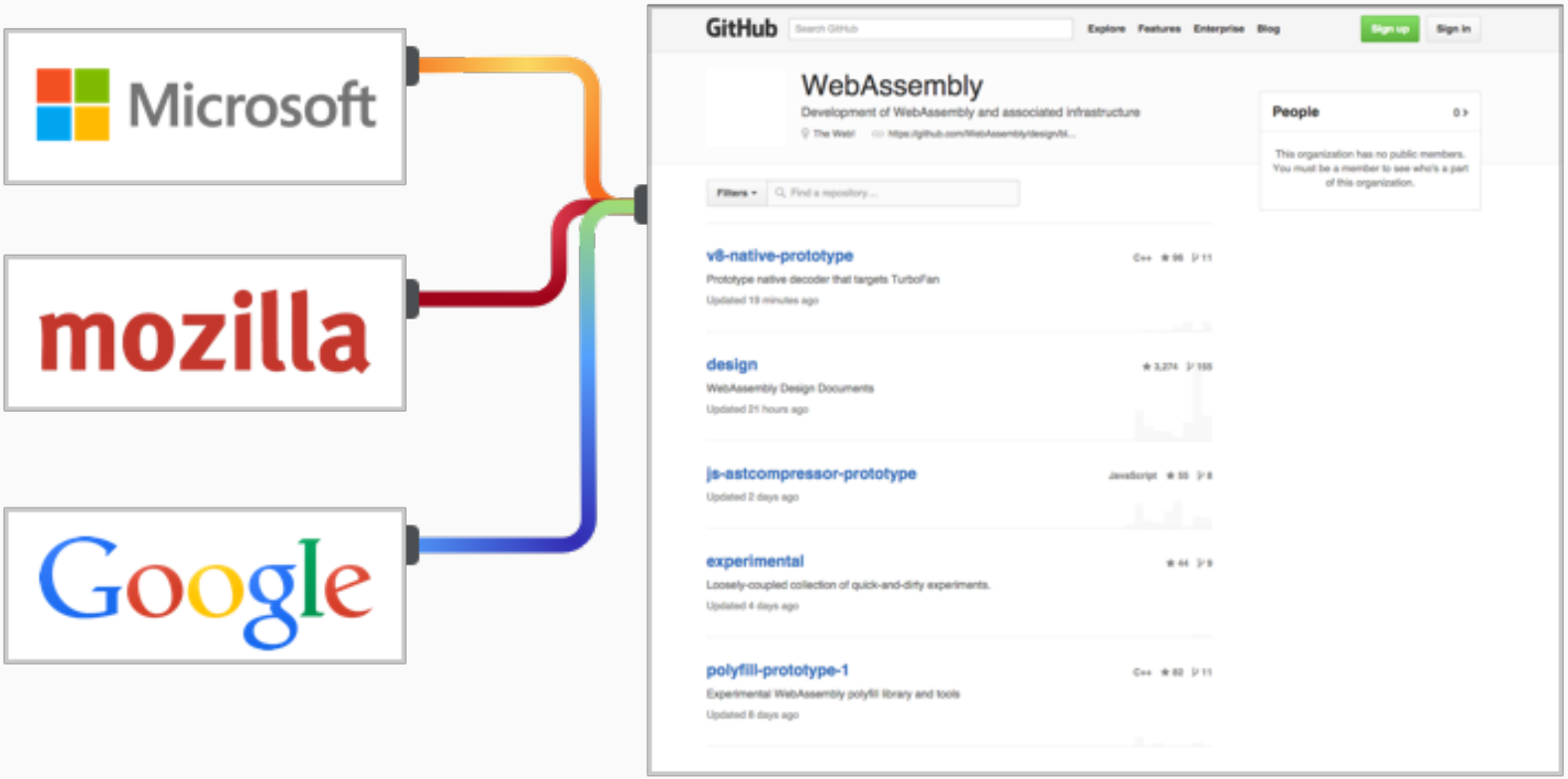


- **Moz://a** Mozilla Foundation
- **Open Source** https://github.com/rust-lang/rust
- **Star** 38.651 **Fork** 6.011 **Contributors** 2.457

O que me levou a me interessar por Rust?

- **Paradigmas:**
  - Estruturada, imperativa, concorrente, funcional, compilada
- **Segurança**
  - Segurança de memória sem **Garbage Collector** ou **Runtime**.
  - Concorrência sem disputa de dados.
- **Performance**
  - Performance comparável a C
  - Abstração sem **overhead**.
- **Casos de sucesso**

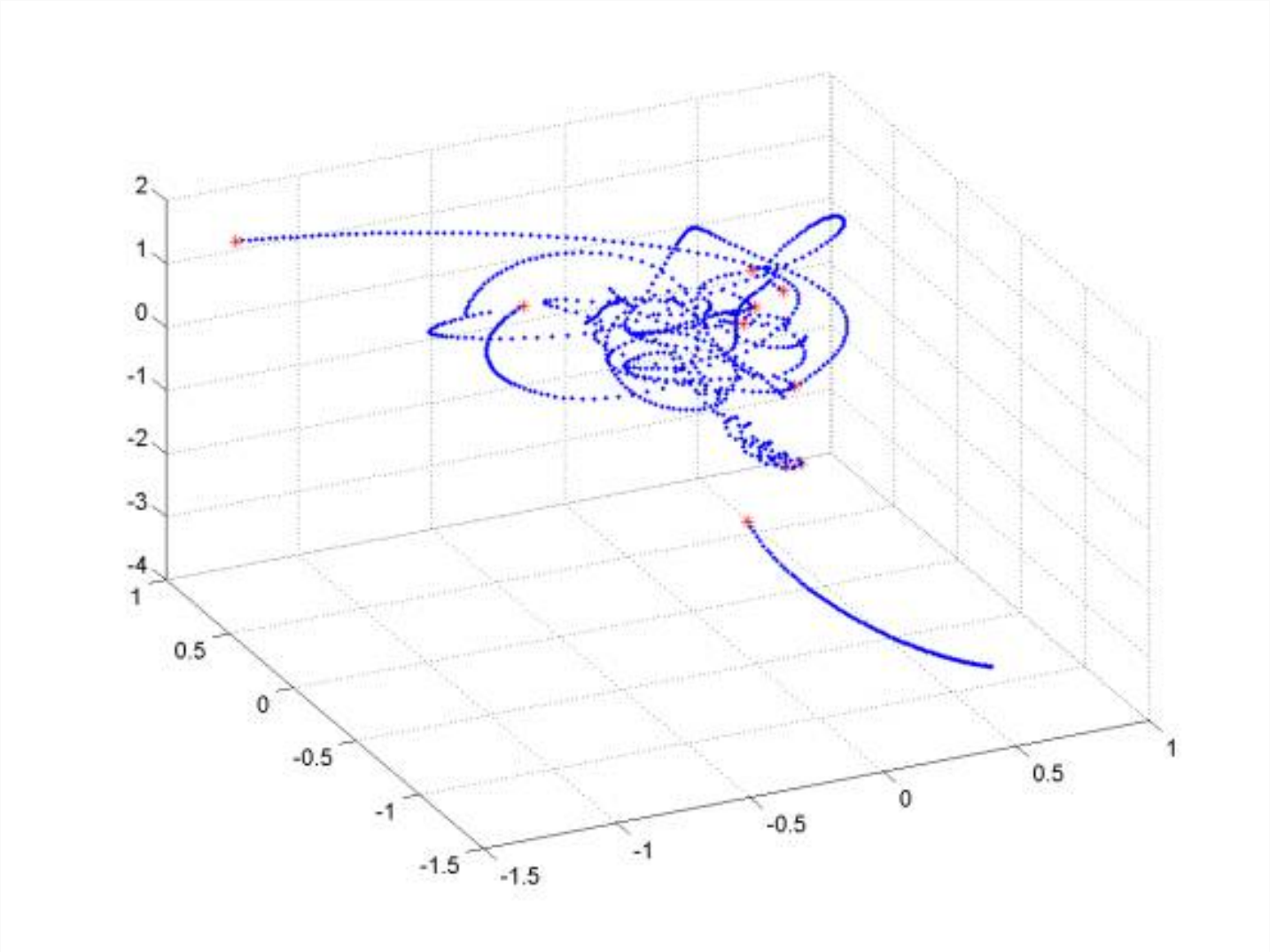
O que me levou a me interessar por Rust?



Possui suporte nativo ao WebAssembly

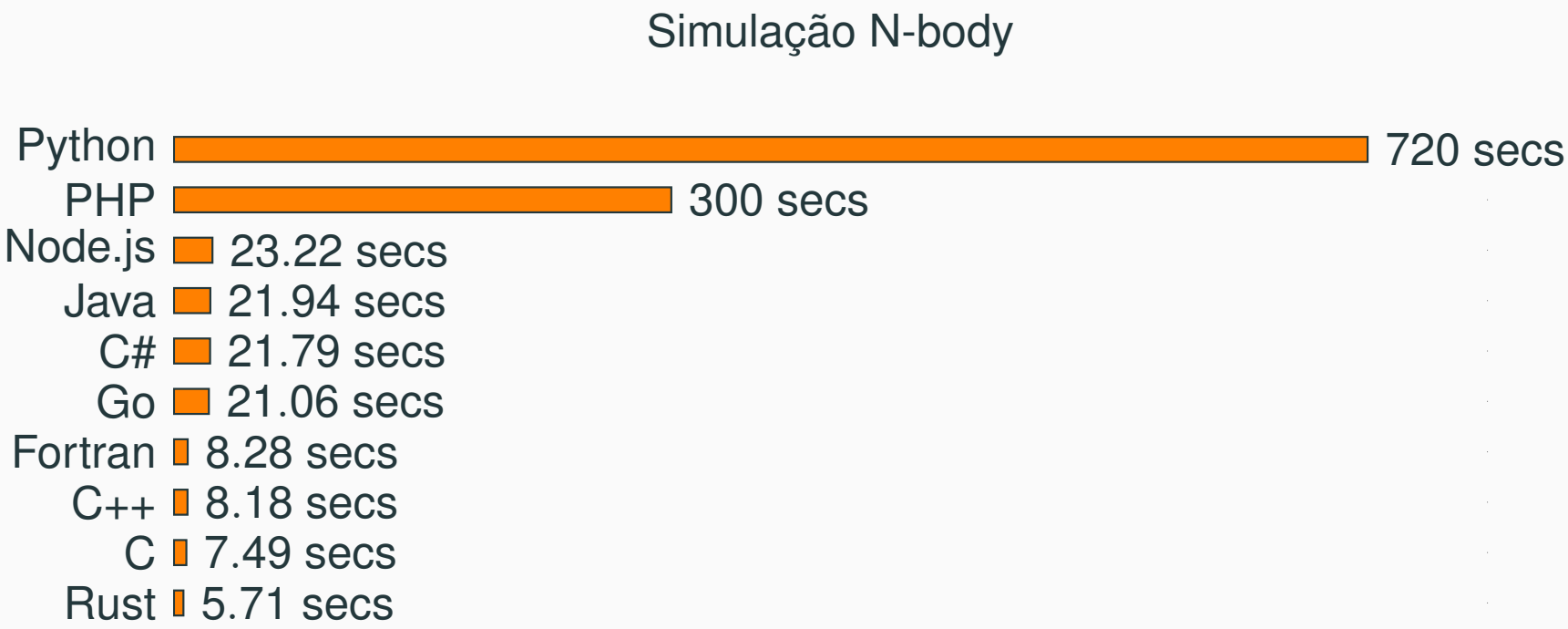
https://webassembly.org/

O que me levou a me interessar por Rust?



N-body permite calcular a evolução gravitacional de uma galáxia

O que me levou a me interessar por Rust?



Simular a evolução gravitacional de 4 planetas com N = 50.000.000

https://benchmarksgame-team.pages.debian.net/benchmarksgame/

Que é Ownership

```
1 fn main() {
2     let s1 = String::from("Hello␣World");
3
4     let _s2 = s1; // <= Ownership
5
6     println!("{}", s1);
7 }
```

```
1 error[E0382]: use of moved value: ‘s1‘
```

- Rust a memória é gerenciada através de um sistema de posse
- Gera **código nativo** inteligente de desalocação.
- Sem os **memory-leaks** das linguagens não gerenciadas.
- Sem os **overheads** de processamento causados pelos **GCs**.

Ownership - Clone ou Referência

```rust
// Ownership
fn main() {
    let s1 = String::from("Hello World");

    // Clona s1 para _s2
    let _s2 = s1.clone();

    // OU

    // referencia ao valor s1, mas nao o possui.
    let _s2 = &s1;

    println!("{}", s1);
}
```

Variáveis e Mutabilidade

```
fn main() {
    let x = 5;

    x = 6;

    println!("O␣valor␣de␣x␣e␣{}", x);
}
```

```
error[E0384]: cannot assign twice to immutable variable ‘x‘
```

- Em Rust, por padrão, as variáveis são **imutáveis**.
- O compilador não deixa atribuir mais de uma vez à variável imutável **x**.

Variáveis e Mutabilidade - mut

```
fn main() {
    let mut x = 5;

    x = 6;

    println!("O valor de x e {}", x);
}
```

```
$ cargo run

O valor de x e: 5
O valor de x e: 6
```

- A palavra chave **mut** autoriza a mudar o valor 5 contido em x para 6.

Shadowing

```
fn main() {
    let x = 5;

    let x = x + 1;

    let x = x * 2;

    println!("O valor de x e: {}", x);
}
```

```
$ cargo run

O valor de x e: 12
```

- Você pode declarar uma nova variável com o mesmo nome de uma variável anterior, e a nova variável sombreia a variável anterior.

Operador match

```
fn fibonacci(n: i32) -> u64 {
    match n {
        0 => panic!("zero␣is␣not␣argument!"),
        1 | 2 => 1,
        50 => 12586269025,
        100 => 354224848179261915075,
        _  => fibonacci(n - 1) + fibonacci(n - 2)
    }
}
```

- O operador **match** nos permite comparar um valor com uma série de padrões e executar um código com base no padrão que casar.

Tratamento de Erros

```rust
enum Result<T, E> {
    Ok(T),
    Err(E),
}
```

- Rust agrupa erros em duas categorias principais:
  - **recuperáveis** - Erros recuperáveis são situações em que é razoável reportar o problema ao usuário.
  - **irrecuperáveis** - Erros irrecuperáveis são sempre sintomas de bugs.

Entrar em panic! ou Não Entrar em panic!

```
use std::fs::File;

fn main() {
    let f = File::open("hello.txt");

    let f = match f {
        Ok(file) => file,
        Err(error) => {
            panic!("fail: {:?}", error)
        },
    };
}
```

- Em algumas situações é mais apropriado escrever código que entra em pânico em vez de retornar um Result.

Esolangs (Linguagens exóticas)

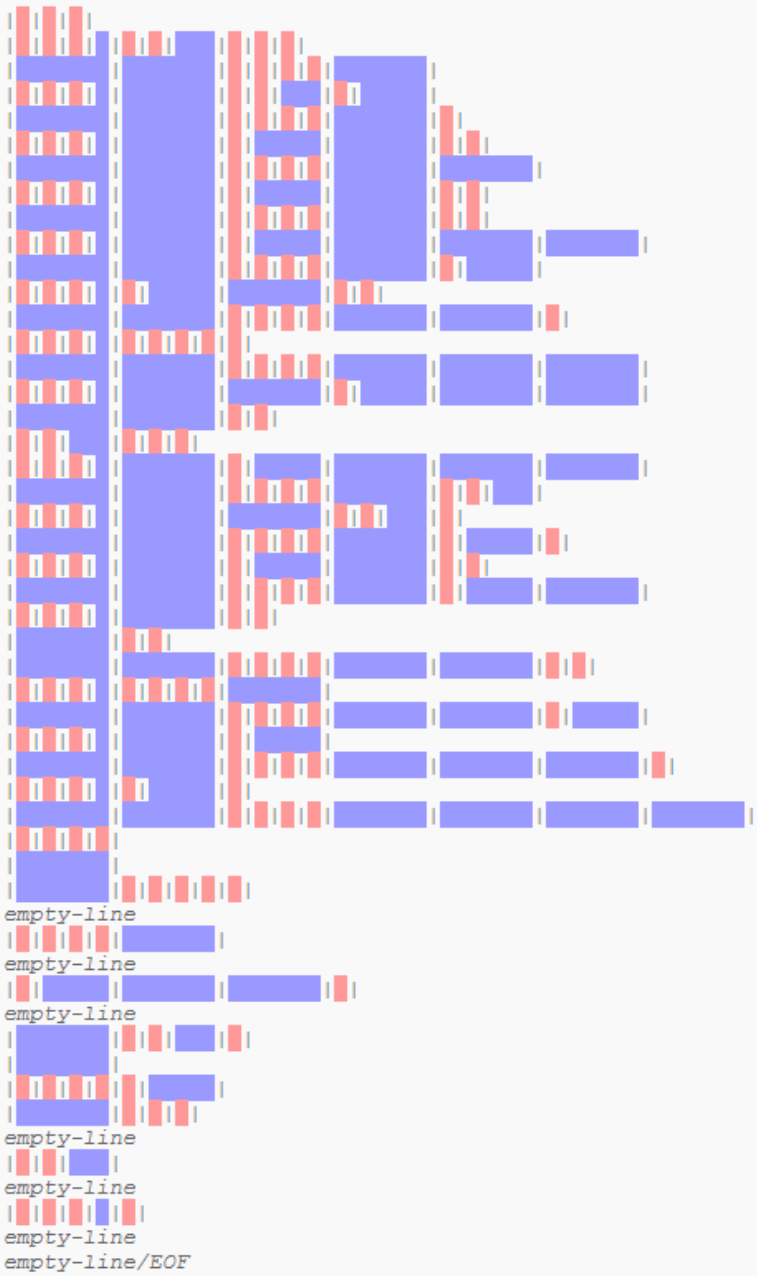
Benfuge (não use)

Benfuge é uma linguagem escrita bidimensionalmente, seu código pode andar em 4 direções.

```
>                  v
v ,,,,,"Hello"<
>48*,            v
v,,,,,,"World!"<
>25*,@
```

https://catseye.tc/article/Languages.md#befunge-93

Whitespace (não use)



BIRL (Bambam's "It's show time"Recursive Language)


```
HORA DO SHOW
    CE QUER VER ESSA PORRA? ("Hello, World!\n");
    BORA CUMPADE 0;
BIRL
```

Baseada em ArnoldC segundo o autor "Deve ser utilizada apenas por quem realmente constrói fibra e não é água com código." (Made in Brazil)
https://birl-language.github.io/

Any questions