

# API Days Paris



Zacaria Chtatar - December 2023

<https://havesome-rust-apidays.surge.sh>

**Forget TypeScript**

**Choose Rust**

**to build**

**Robust, Fast and Cheap APIs**

# Some context

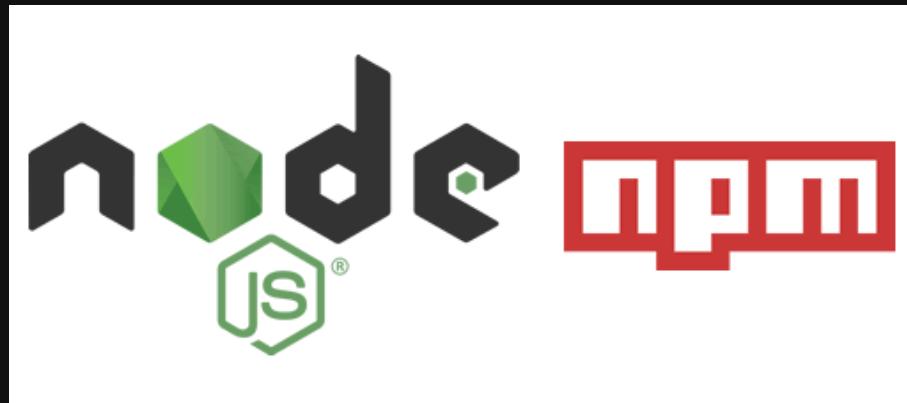
# 2009

JS lives in the browser



# Until Nodejs

V8 engine, modules system & NPM



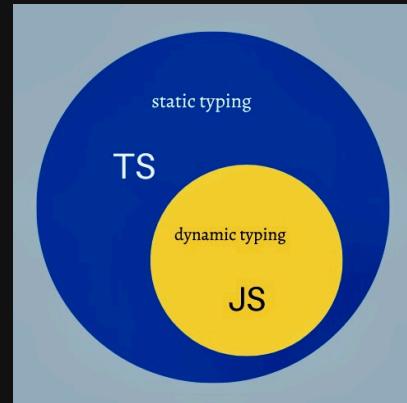
- modules
- boom of easy to reuse code
- fullstack JS : Easy to start with, hard to master

# Fullstack

- paradigm clash between static and dynamic
- JS doesn't help you follow strict API interfaces

```
✖ ► TypeError: Cannot read properties of undefined (reading '1')
  at main.fe86f8ba.chunk.js:1
  at Array.map (<anonymous>)
  at main.fe86f8ba.chunk.js:1
  at Object.useMemo (2.632b68e4.chunk.js:2)
  at t.useMemo (2.632b68e4.chunk.js:2)
  at et (main.fe86f8ba.chunk.js:1)
  at ia (2.632b68e4.chunk.js:2)
  at Vs (2.632b68e4.chunk.js:2)
  at Cl (2.632b68e4.chunk.js:2)
  at Sl (2.632b68e4.chunk.js:2)
```

# TypeScript



## Benefits:

- IDE Developer experience
- OOP patterns
- compiler checks
- type system

=> better management of big codebase

# Pain points

- does not save you from dealing with JS
- adds types management
- adds static layer onto dynamic layer

JSDoc answers to the precise problem of hinting types without getting in  
your way

**New stakes, new needs**

## Stakes

---

worldwide scale

privacy

market competition

environment

human lives

## Needs

---

scalability

security

functionality

computation time

memory footprint

safety

**TypeScript is not enough**

# Introducing Rust

*Fast, Reliable, Productive: pick three*

# Stable



I call it my billion-dollar mistake. It  
was the invention of the null  
reference in 1965.

— *Tony Hoare* —

AZ QUOTES

# enums

```
struct FakeCat {  
    alive: bool,  
    hungry: bool,  
}  
  
let zombie = FakeCat { alive: false, hungry: true }; // ???
```

*Rust makes it easy to make invalid state  
unrepresentable*

```
enum RealCat {  
    Alive { hungry: bool }, // enums can contain structs  
    Dead,  
}  
  
let cat = RealCat::Alive { hungry: true };  
let dead_cat = RealCat::Dead;
```

# Option enum

```
// full code of the solution to the billion dollar mistake
// included in the standard library
enum Option<T> {
    None,
    Some(T),
}
```

```
let item: Option<Item> = get_item();

match item {
    Some(item) => map_item(item),
    None => handle_none_case(), // does not compile if omitted
}
```

# What is wrong with this function ?

```
function getConfig(path: string): string {  
  return fs.readFileSync(path);  
}
```

There is no hint that `readFileSync` can throw an error under some conditions

# Result enum

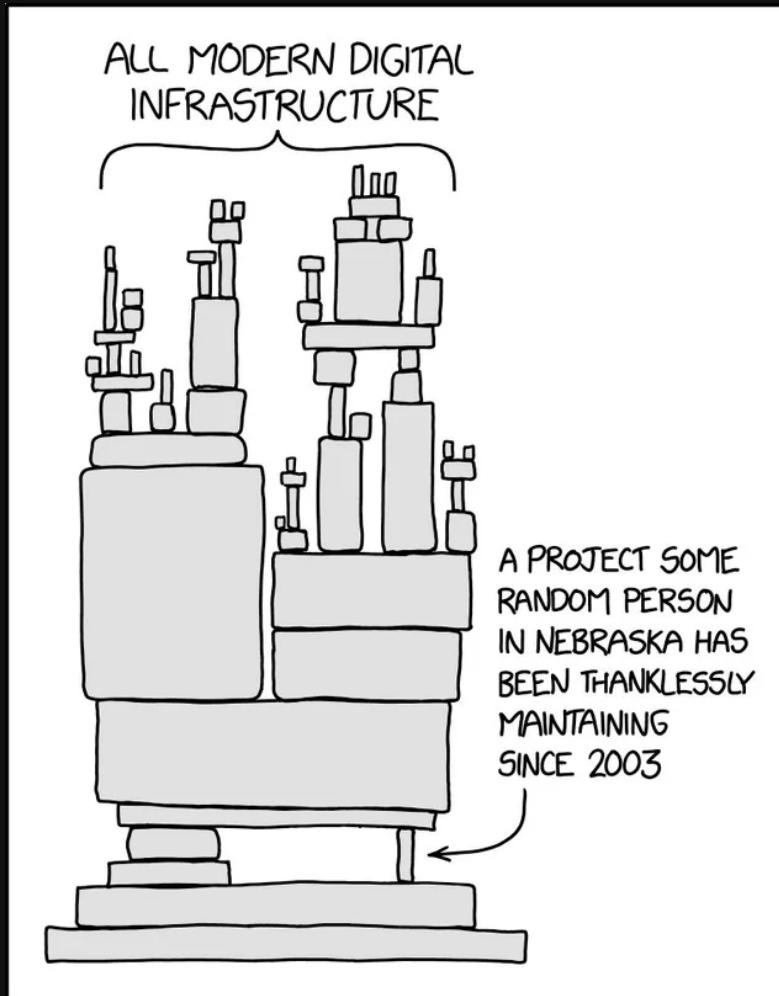
```
enum Result<T, E> {  
    Ok(T),  
    Err(E),  
}
```

```
fn get_config(path: &str) -> Result<String, io::Error> {  
    fs::read_to_string(path)  
}  
  
// That's a shortcut, don't send me to prod !  
fn dirty_get_config(path: &str) -> String {  
    fs::read_to_string(path).unwrap() // panics in case of error  
}
```

We need to clarify what can go wrong

It's even better when it's embedded in the language

## 2016 : Do you remember the left-pad incident ?



Source

# crates.io

- no crate (package) unpublish
- can disable crate only for new projects

# Linux: The Kernel



- attract young devs
- 2/3 of vulnerabilities come from memory management
- Kernel is in C and Assembly
- Linus Torvalds : C++ 

# Fast

Total			
	Energy	Time	
(c) C	1.00	(c) C	1.00
<b>(c) Rust</b>	<b>1.03</b>	<b>(c) Rust</b>	<b>1.04</b>
(c) C++	1.34	(c) C++	1.56
(c) Ada	1.70	(c) Ada	1.85
(v) Java	1.98	(v) Java	1.89
(c) Pascal	2.14	(c) Chapel	2.14
(c) Chapel	2.18	(c) Go	2.83
(v) Lisp	2.27	(c) Pascal	3.02
(c) Ocaml	2.40	(c) Ocaml	3.09
(c) Fortran	2.52	(v) C#	3.14
(c) Swift	2.79	(v) Lisp	3.40
(c) Haskell	3.10	(c) Haskell	3.55
(v) C#	3.14	(c) Swift	4.20
(c) Go	3.23	(c) Fortran	4.20
(i) Dart	3.83	(v) F#	6.30
(v) F#	4.13	<b>(i) JavaScript</b>	<b>6.52</b>
<b>(i) JavaScript</b>	<b>4.45</b>	(i) Dart	6.67
<b>(v) Racket</b>	<b>7.91</b>	(v) Racket	11.27
<b>(i) TypeScript</b>	<b>21.50</b>	(i) Hack	26.99
(i) Hack	24.02	(i) PHP	27.64
(i) PHP	29.30	(v) Erlang	36.71
(v) Erlang	42.23	(i) Jruby	43.44
(i) Lua	45.98	<b>(i) TypeScript</b>	<b>46.20</b>
(i) Jruby	46.54	(i) Ruby	59.34
(i) Ruby	69.91	(i) Perl	65.79
(i) Python	75.88	(i) Python	71.90
(i) Perl	79.58	(i) Lua	82.91

2017 : Energy efficiency accross programming languages

# Github: Code Search index

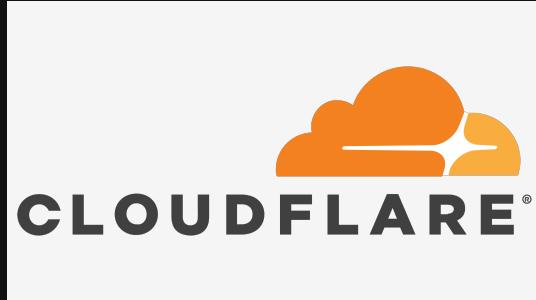


45 million repos

28 TB of unique content

- several months with Elasticsearch
- 36h with Rust and Kafka
- 640 queries /s

# Cloudflare: HTTP proxy



- nginx not fast enough 😱
- hard to customize in C
- allows to share connections between threads

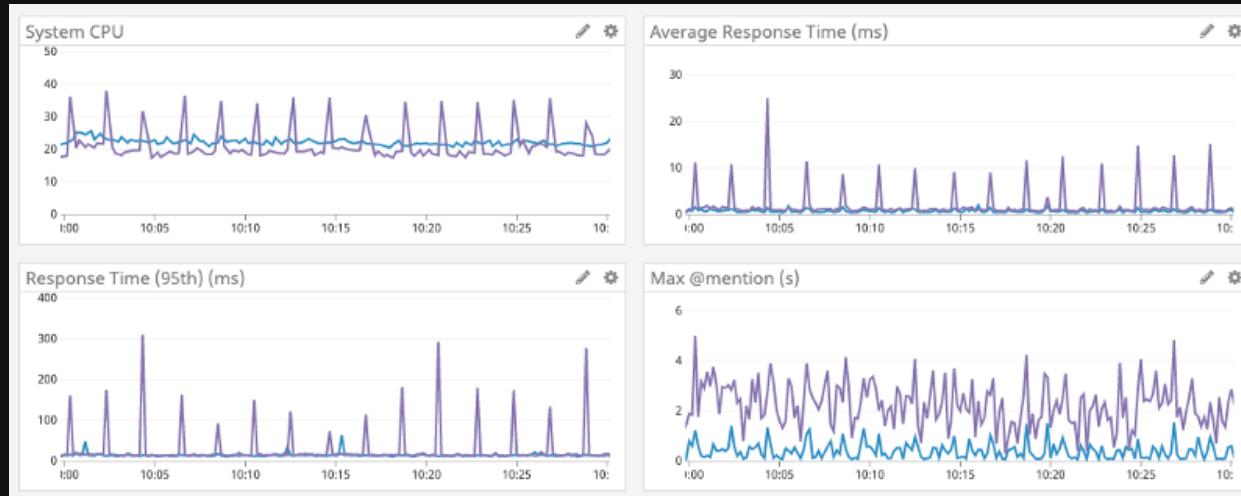
= 160x less connections to the origins

= 434 years less handshakes per day

# Discord: Message read service



- cache of a few billion entries
- every connection, message sent and read...
- latencies every 2 minutes because of Go Garbage Collector



Cheap

Total			
	Energy	Time	
		Mb	
(c) C	1.00	(c) C	1.00
<b>(c) Rust</b>	<b>1.03</b>	<b>(c) Rust</b>	<b>1.04</b>
(c) C++	1.34	(c) C++	1.56
(c) Ada	1.70	(c) Ada	1.85
(v) Java	1.98	(v) Java	1.89
(c) Pascal	2.14	(c) Chapel	2.14
(c) Chapel	2.18	(c) Go	2.83
(v) Lisp	2.27	(c) Pascal	3.02
(c) Ocaml	2.40	(c) Ocaml	3.09
(c) Fortran	2.52	(v) C#	3.14
(c) Swift	2.79	(v) Lisp	3.40
(c) Haskell	3.10	(c) Haskell	3.55
(v) C#	3.14	(c) Swift	4.20
(c) Go	3.23	(c) Fortran	4.20
(i) Dart	3.83	(v) F#	6.30
(v) F#	4.13	<b>(i) JavaScript</b>	<b>6.52</b>
<b>(i) JavaScript</b>	<b>4.45</b>	(i) Dart	6.67
(v) Racket	7.91	(v) Racket	11.27
<b>(i) TypeScript</b>	<b>21.50</b>	(i) Hack	26.99
(i) Hack	24.02	(i) PHP	27.64
(i) PHP	29.30	(v) Erlang	36.71
(v) Erlang	42.23	(i) Jruby	43.44
(i) Lua	45.98	<b>(i) TypeScript</b>	<b>46.20</b>
(i) Jruby	46.54	(i) Ruby	59.34
(i) Ruby	69.91	(i) Perl	65.79
(i) Python	75.88	(i) Python	71.90
(i) Perl	79.58	(i) Lua	82.91

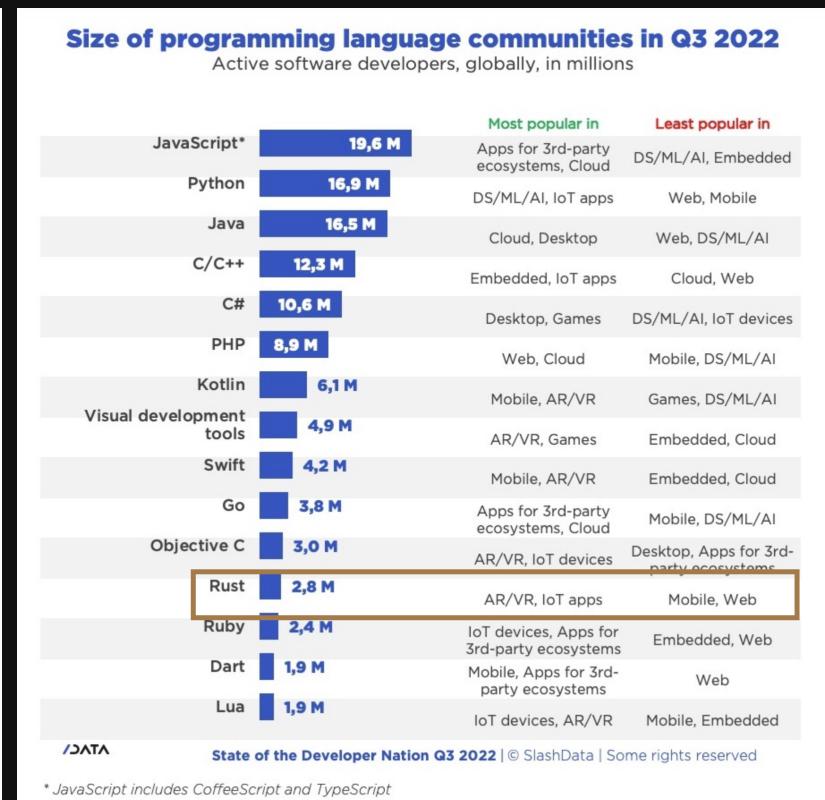
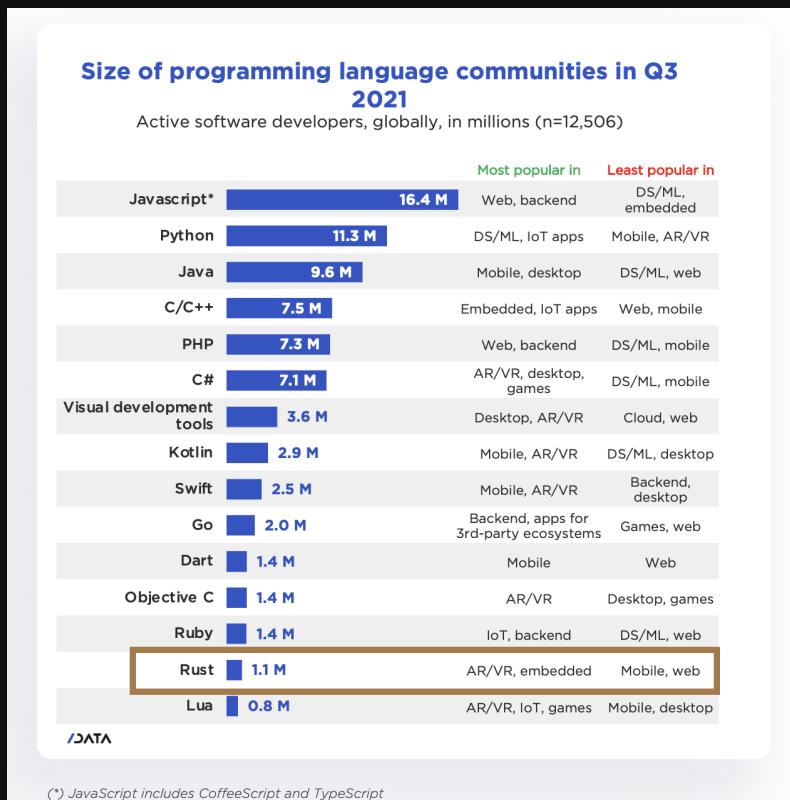
- cpu & memory
- less bugs
- learning curve
- less cases to test

# Attractive

Most admired language according to StackOverflow for 8 years !

Only place where there is **more** devs available than offers

# Growing community



# Features

- static types
- compiled
- no GC
- compiler developed in Rust
- low and high level : zero cost abstraction
- no manual memory management : Ownership & Borrow checker

=> There is no blackbox between you and the machine

=> Better predictability

=> Awesome developer experience

# Ownership rules

- Only one variable owns data at a time
- Multiple readers or one writer

=> Memory is freed as soon as variable is out of scope

**It's like a fundamental problem has been solved**

# The compiler ❤

```
fn say(message: String) {  
    println!("{}", message);  
}  
  
fn main() {  
    let message = String::from("hey");  
    say(message);  
    say(message);  
}
```

```
1 error[E0382]: use of moved value: `message`  
--> src/main.rs:12:9  
10 |     let message = String::from("hey");  
   |           ----- move occurs because `message` has type `String`, which does  
   |           not implement the `Copy` trait  
11 |     say(message);  
   |           ----- value moved here  
12 |     say(message);  
   |           ^^^^^^^ value used here after move  
  
note: consider changing this parameter type in function `say` to borrow instead  
if owning the value isn't necessary  
--> src/main.rs:5:17  
5  fn say(message: String) {  
   |           ^^^^^^ this parameter takes ownership of the value  
   |           in this function  
help: consider cloning the value if the performance cost is acceptable  
11  say(message.clone());  
   +++++++  
  
For more information about this error, try `rustc --explain E0382`.  
error: could not compile `learn-dyn` (bin "learn-dyn") due to previous error
```

# Tools

- cargo test : Integration Tests, Unit Tests
- cargo fmt
- cargo bench
- clippy : lint
- bacon : reload
- rust-analyzer : IDE developer experience

# Cargo doc stays up to date

```
1 /// Formats the sum of two numbers as a string.  
2 ///  
3 /// # Examples  
4 ///  
5 /// ``  
6 /// let result = mycrate::sum_as_string(5, 10);  
7 /// assert_eq!(result, "15");  
8 /// ``  
9 pub fn sum_as_string(a: i32, b: i32) -> String { (a + b).to_string() }
```

cargo doc --open

# serde\_json

1.0.108

All Items

Modules

Macros

Structs

Enums

Functions

Type Aliases

Crates

serde\_json

Click or press 'S' to search, '?' for more options...



## Create serde\_json

source · [-]

### [+] Serde JSON

JSON is a ubiquitous open-standard format that uses human-readable text to transmit data objects consisting of key-value pairs.

```
{  
    "name": "John Doe",  
    "age": 43,  
    "address": {  
        "street": "10 Downing Street",  
        "city": "London"  
    },  
    "phones": [  
        "+44 1234567",  
        "+44 2345678"  
    ]  
}
```

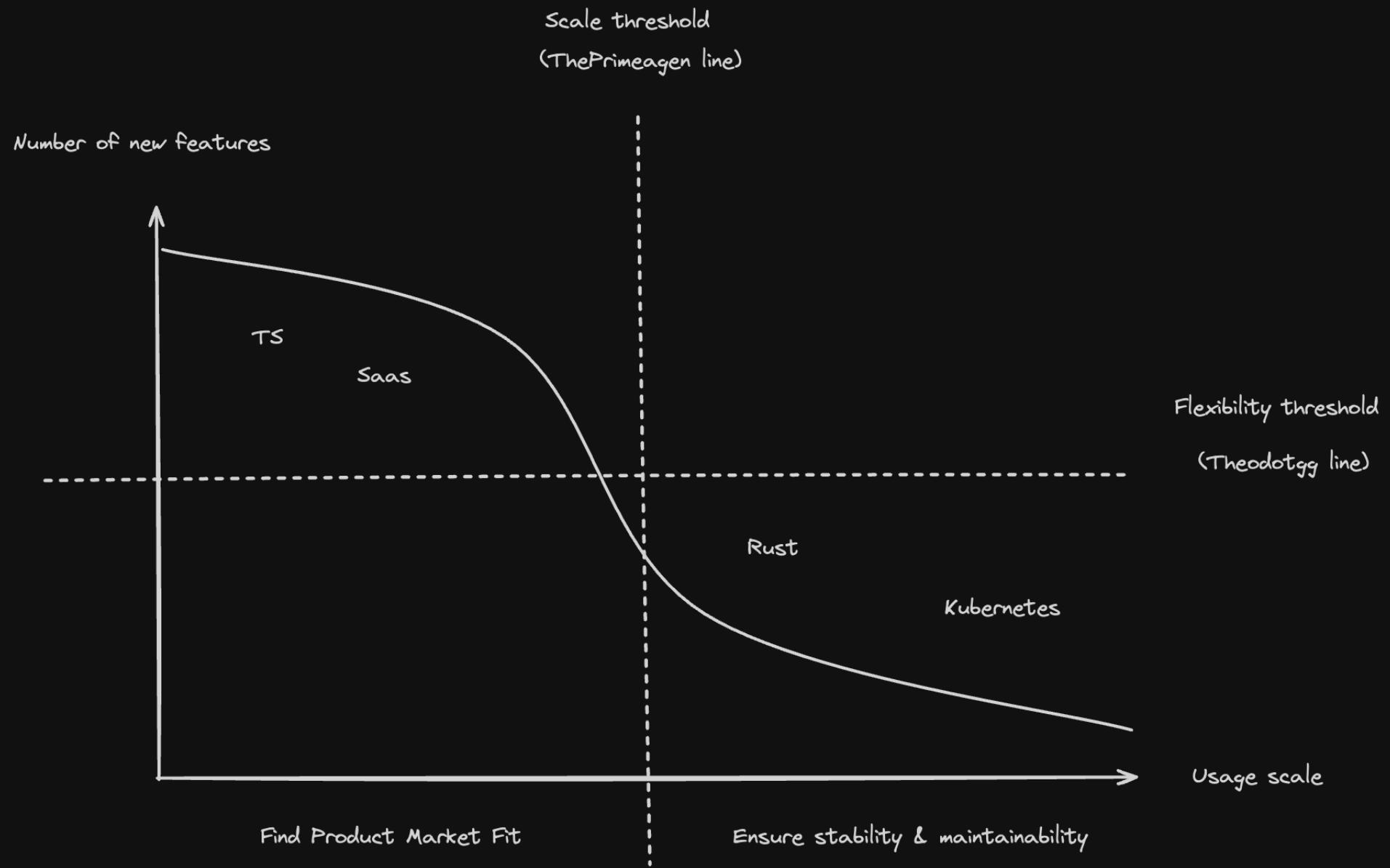
There are three common ways that you might find yourself needing to work with JSON data in Rust.

- **As text data.** An unprocessed string of JSON data that you receive on an HTTP endpoint, read from a file, or prepare to send to a remote server.
- **As an untyped or loosely typed representation.** Maybe you want to check that some JSON data is valid before

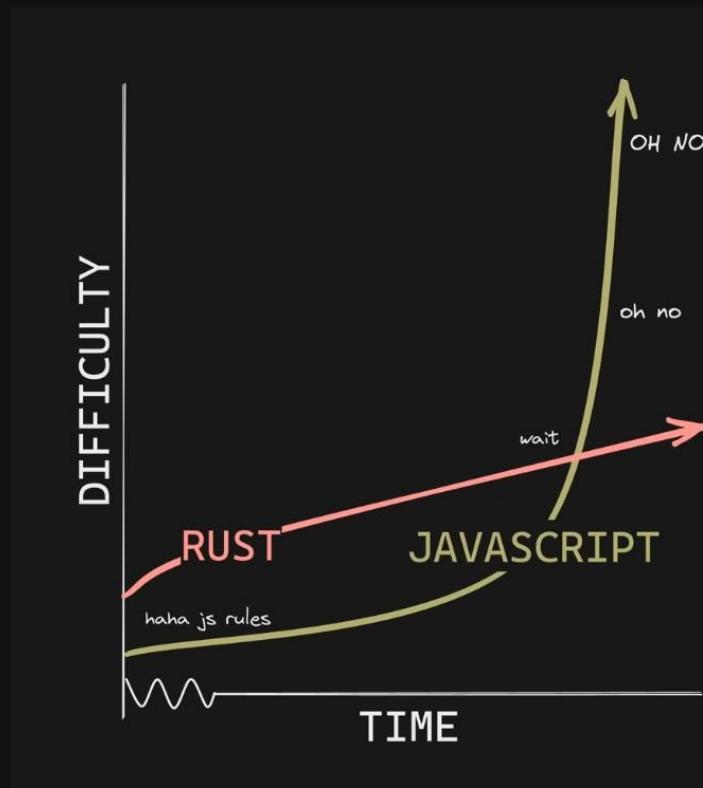
# Possible struggles

- projects move slower
- embrace the paradigm
- takes 3 to 6 months to become productive
- build time
- work with external libraries
- ecosystem calmer than JS

# Project lifetime



*In other languages simple things are easy and complex things are possible, in Rust simple things are possible and complex things are EASY.*



# Get started as dev

- Rust book - reference
- Rust by example - condensed reference
- Rustlings - exercises
- A half-hour to learn Rust - quick walkthrough
- Comprehensive Rust by Google - complete walkthrough
- Noboilerplate - quick videos ❤️
- Code to the moon - longer videos
- Let's get rusty- Youtube & paid bootcamp - not sponsored
- Awesome Rust - keywords discovery
- Roadmap - keywords discovery

# Get started as manager

- find dev interested in Rust: there are a lot
- start with simple projects:
  - CLI
  - lambdas
  - microservice
  - network app
  - devops tools

Make the world a safer, faster and sustainable place

# Thank you



Slides : <https://havesome-rust-apidays.surge.sh>

X @ChtatarZacaria - 🌎 [havesomecode.io](https://havesomecode.io)