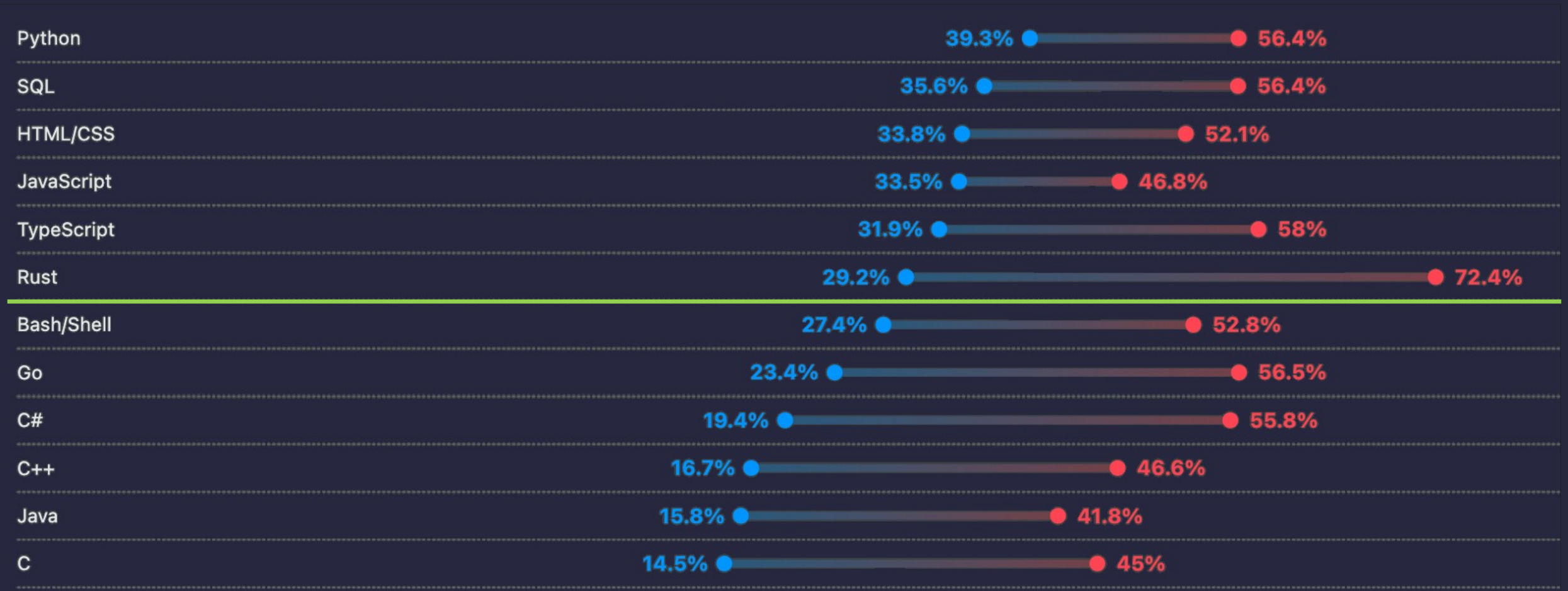
RUST

(FOR GIS)

Padányi-Gulyás Gergely | Foszforgézu 2026.

THE MOST LOVED LANGUAGE

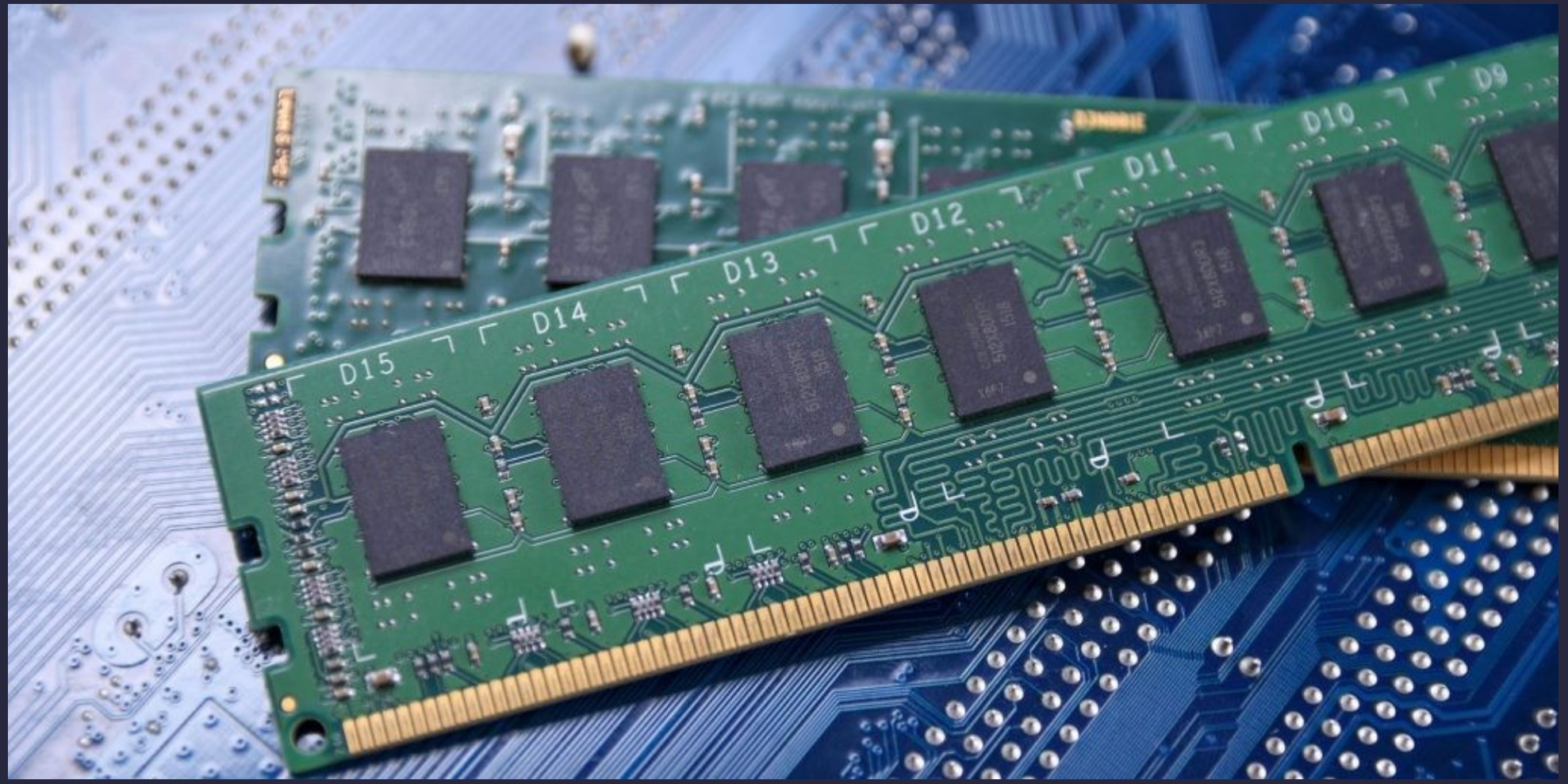


https://survey.stackoverflow.co/2025/technology#admired-and-desired


Graydon
Hoare

“It’s ridiculous, that we computer people couldn’t even make an elevator that works without crashing!”

MEMORY MANAGEMENT



MANUAL

MANUAL

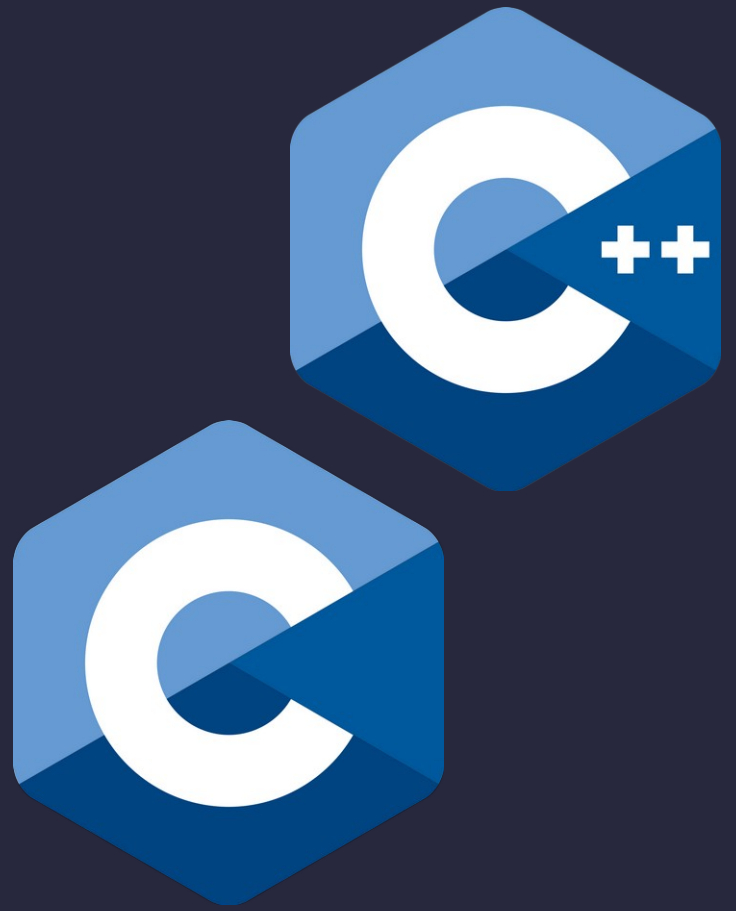
MANUAL

malloc()

calloc()

realloc()

free()



MANUAL

malloc()

free()

realloc()

calloc()

Fine-grained control over memory

MANUAL

malloc()

free()

realloc()

calloc()

Fine-grained control over memory

70% of the vulnerabilities are due to memory errors

GARBAGE COLLECTOR

GARBAGE COLLECTOR

GARBAGE COLLECTOR

Simplifies development

GARBAGE COLLECTOR

Simplifies development

Runtime overhead

Unpredictable pause times

MANUAL VS. GC

Programmers manage memory.
Software is fast.

Runtime manages memory.
Software is safe.

Rust tries to make
fast software and
safe software the
same thing.

Rust moves many bugs from production incidents to compiler errors.

Memory is managed through a system of ownership with a set of rules that the compiler checks. If any of the rules are violated, the program won't compile.

None of the features of ownership will slow down your program while it’s running.

```rust
fn main() {
    let conference_title: String = String::from("FOSS4G.hu");
    let year: i32 = 2026;

    greet(conference_title, year);
}

fn greet(title: String, year: i32) {
    println!("Welcome to {title} in {year}.");
}
```

```
foss4g_hu on  main [?] via 🦀 v1.89.0
> cargo run
   Compiling foss4g_hu v0.1.0 (/Users/padanyig/code/foss4g_hu)
    Finished `dev` profile [unoptimized + debuginfo] target(s) in 0.20s
     Running `target/debug/foss4g_hu`
Welcome to FOSS4G.hu in 2026.
```

OWNERSHIP & BORROWING

OWNERSHIP

- Each value in Rust has an owner.
- There can only be one owner at a time.
- When the owner goes out of scope, the value will be dropped.

BORROWING

- There can be any number of immutable references (&T) to a resource.
- There can be exactly one mutable reference (&mut T).

```rust
fn main() {
    let conference_title: String = String::from("FOSS4G.hu");
    let year: i32 = 2026;

    greet(conference_title, year);


}

fn greet(title: String, year: i32) {
    println!("Welcome to {title} in {year}.");
}
```

```rust
fn main() {
    let conference_title: String = String::from("FOSS4G.hu");
    let year: i32 = 2026;

    greet(conference_title, year);

    println!("The conference title is {conference_title}.");
}

fn greet(title: String, year: i32) {
    println!("Welcome to {title} in {year}.");
}
```


fn main() {
let conference_title: String = String::from("FOSS4G.hu");
let year: i32 = 2026;
greet(conference_title, year);
value moved here
println!("The conference title is {conference_title}.");
}
fn greet(title: String, year: i32) {
consider changing this
println!("Welcome to {title} in {year}.");
}
Not Committed Yet

```
error[E0382]: borrow of moved value: `conference_title`
  ⟶ src/main.rs:7:40
   |
2  |     let conference_title: String = String::from("FOSS4G.hu");
   |         ---------------- move occurs because `conference_title` has type `String`, which does
 not implement the `Copy` trait
...
5  |     greet(conference_title, year);
   |           ---------------- value moved here
6  |
7  |     println!("The conference title is {conference_title}.");
   |                                       ^^^^^^^^^^^^^^^^^ value borrowed here after move
   |
note: consider changing this parameter type in function `greet` to borrow instead if owning the va
lue isn't necessary
  ⟶ src/main.rs:10:17
   |
10 | fn greet(title: String, year: i32) {
   |    -----        ^^^^^^ this parameter takes ownership of the value
   |    |
   |    in this function
   = note: this error originates in the macro `$crate::format_args_nl` which comes from the expans
ion of the macro `println` (in Nightly builds, run with -Z macro-backtrace for more info)
```

```rust
fn main() {
    let conference_title: String = String::from("FOSS4G.hu");
    let year: i32 = 2026;

    greet( conference_title, year);

    println!("The conference title is {conference_title}.");
}

fn greet(title:  String, year: i32) {
    println!("Welcome to {title} in {year}.");
}
```

```rust
fn main() {
    let conference_title: String = String::from("FOSS4G.hu");
    let year: i32 = 2026;

    greet(&conference_title, year);

    println!("The conference title is {conference_title}.");
}

fn greet(title: &String, year: i32) {
    println!("Welcome to {title} in {year}.");
}
```

```rust
fn main() {
    let conference_title: String = String::from("FOSS4G.hu");
    let year: i32 = 2026;

    greet(&conference_title, year);

    println!("The conference title is {conference_title}.");
}

fn greet(title: &String, year: i32) {
    println!("Welcome to {title} in {year}.");
}
```

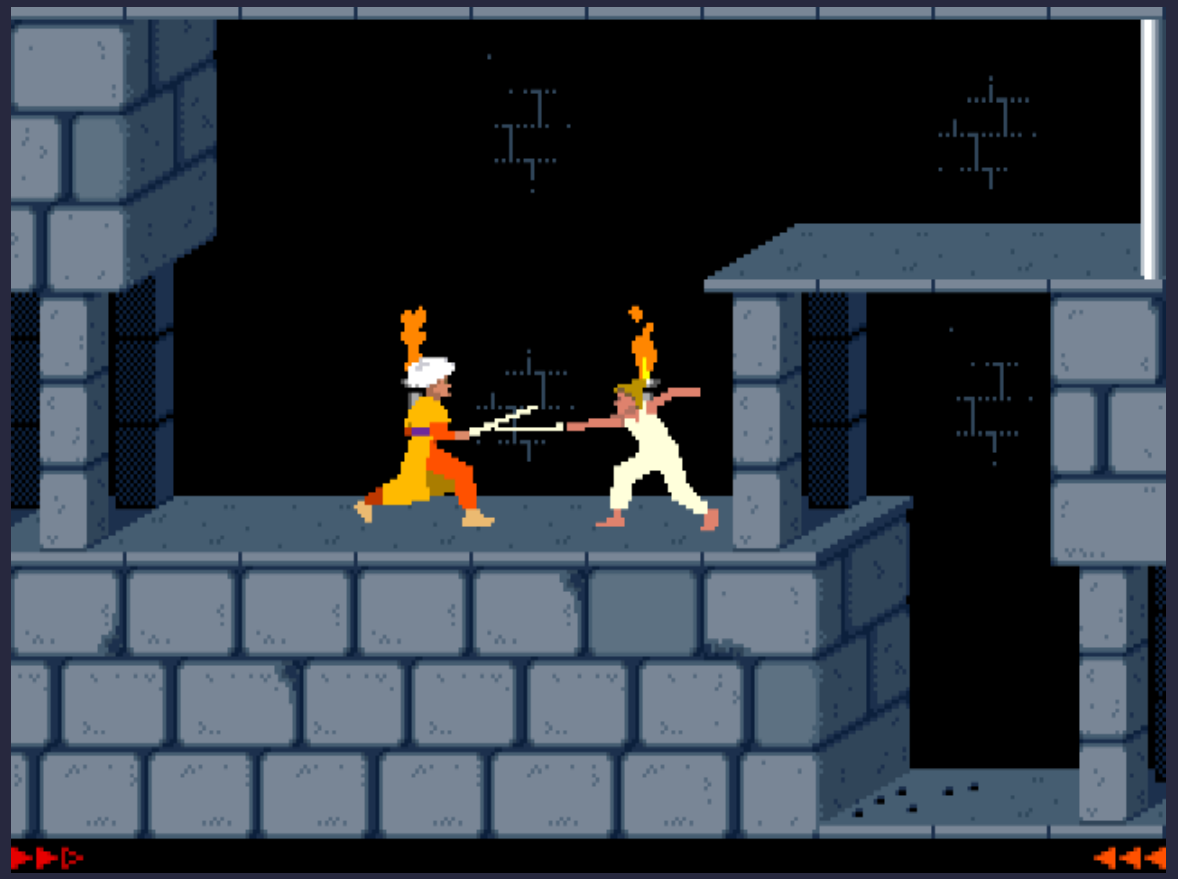
```
foss4g_hu on  main [?] via 🦀 v1.89.0
> cargo run
    Finished `dev` profile [unoptimized + debuginfo] target(s) in 0.01s
     Running `target/debug/foss4g_hu`
Welcome to FOSS4G.hu in 2026.
The conference title is FOSS4G.hu.
```

THE TRADE-OFF

- One needs to learn ownership & borrowing

THE TRADE-OFF

- One needs to learn ownership & borrowing
- Fighting with the compiler



THE WIN

- Better software
  - memory safety
  - concurrency safety
  - blazingly fast

THE WIN

- Better software
  - memory safety
  - concurrency safety
  - blazingly fast
- Better programmer

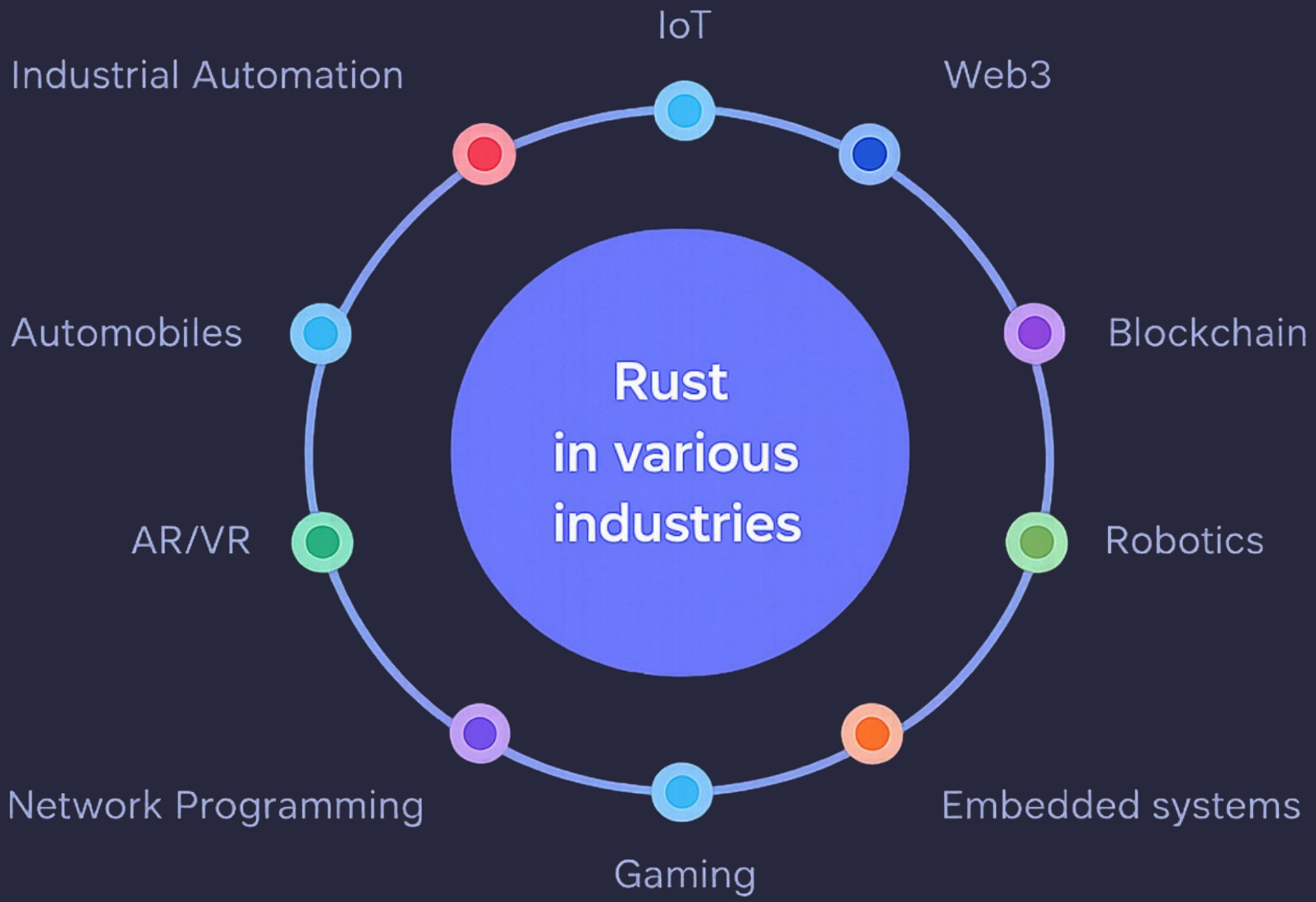
> **"It's *enjoyable* to write Rust, which is maybe kind of weird to say, but it's just the language is fantastic. It's fun. You feel like a magician, and that never happens in other languages."**
>
> Parker Timmerman, software engineer

THE WIN

At Discord, engineers had long been annoyed that the garbage collector in Go would slow things down. Their Go software would carry out the procedure roughly every two minutes, even though the Discord engineers had written things so carefully there was no garbage to be collected. In 2020, they rewrote that system in **Rust**, and discovered it now ran 10 times faster.

THE WIN

At AWS, a study of Rust-based code found it runs so efficiently that it uses half as much electricity as a similar program written in Java, a language commonly used at AWS. “We can do the same work in a data center that’s half the size, letting you tuck one into a city instead of planting it in an exurban field.”


IoT
Industrial Automation
Web3
Automobiles
Blockchain
Rust
in various
industries
AR/VR
Robotics
Network Programming
Embedded systems
Gaming

WHAT ABOUT GIS?

https://

Primitives

Geo	Geospatial primitives such as Point & LineString, and algorithms such as distance, convex hull, centroid calculations.	GitHub	crates.io
Geo-Index	Packed, immutable, zero-copy spatial indexes.	GitHub	crates.io
GEOS	Bindings for the Geometry Engine - Open Source (GEOS) library.	GitHub	crates.io
PROJ	Bindings for the PROJ library for coordinate transformation and projections.	GitHub	crates.io
Robust	Robust primitives for computational geometry.	GitHub	crates.io
Rstar	A spatial index using an R*-tree.	GitHub	crates.io

GIS data formats

GDAL	Bindings for the Geographic Data Abstraction Library (GDAL) for reading and writing raster and vector GIS files.	GitHub	crates.io
Geo-PostGIS	Conversion between geo-types and PostGIS types.	GitHub	crates.io
GeoJSON	Work with GeoJSON files.	GitHub	crates.io
GeoTIFF	Work with GeoTIFF raster files.	GitHub	crates.io
GPX	Work with GPS files.	GitHub	crates.io
KML	Work with KML files.	GitHub	crates.io
netCDF	Bindings for Network Common Data Form (netCDF) library. Can read and write HDF5 files.	GitHub	crates.io
OGC API	OGC API building blocks	GitHub	crates.io

GIS data formats

OSM	Work with OpenStreetMap PBF files.	GitHub	crates.io
PDAL	Bindings for PDAL point-cloud processing library.	GitHub	crates.io
TileJSON	Work with TileJSON files.	GitHub	crates.io
TopoJSON	Work with TopoJSON files.	GitHub	crates.io
Transit	Work with GTFS files.	GitHub	crates.io
WKB	Work with Well-Known Binary (WKB) buffers.	GitHub	crates.io
WKT	Work with Well-Known Text (WKT) files.	GitHub	crates.io
World-file	Work with World-files.	GitHub	crates.io

Utilities

Earcut	Polygon triangulation library.	GitHub	crates.io
Geocoding	Enrich addresses, cities, countries with geographic coordinates through third-party geocoding web services.	GitHub	crates.io
GeographicLib	A Rust port of GeographicLib.	GitHub	crates.io
GeoHash	Compute geohash of locations.	GitHub	crates.io
Geo-SVG	Generate SVGs for geo-types	GitHub	crates.io
GeoZero	Zero-Copy reading and writing of geospatial data.	GitHub	crates.io
Polyline	Encode and decode addresses using the Google Polyline format.	GitHub	crates.io

GEOS BINDINGS

```rust
use geos::Geom;

fn main() {
    let g1 = geos::Geometry::new_from_wkt("POLYGON ((0 0, 0 5, 5 5, 5 0, 0 0))")?;
    let g2 = geos::Geometry::new_from_wkt("POLYGON ((1 1, 1 3, 5 5, 5 0, 1 1))")?;

    let pg1 = geos::PreparedGeometry::new(&g1)?;
    assert_eq!(pg1.intersects(&g2)?, true);
}
```

PROJ BINDINGS

```rust
use proj::Proj;

fn main() {
    let from = "EPSG:2230";
    let to = "EPSG:26946";
    let ft_to_m = Proj::new_known_crs(&from, &to, None).unwrap();
    let result = ft_to_m
        .convert((4760096.421921f64, 3744293.729449f64))
        .unwrap();
    assert_eq!(result.0, 1450880.2910605003);
    assert_eq!(result.1, 1141263.0111604529);
}
```

GDAL BINDINGS

```rust
use gdal::raster::RasterBand;
use gdal::{Dataset, Metadata};
use std::path::Path;

fn main() {
    let path = Path::new("./fixtures/tinymarble.tif");
    let dataset = Dataset::open(path).unwrap();
    println!("dataset description: {:?}", dataset.description());

    let rasterband: RasterBand = dataset.rasterband(1).unwrap();
    println!("rasterband no_data_value: {:?}", rasterband.no_data_value());
    println!("rasterband type: {:?}", rasterband.band_type());
    println!("rasterband offset: {:?}", rasterband.offset());
    if let Ok(rv) = rasterband.read_as::<u8>((20, 30), (2, 3), (2, 3), None) {
        println!("{:?}", rv.data());
    }
}
```

LINKS

- https://rust-lang.org
- https://doc.rust-lang.org/book
- https://www.udemy.com/course/learn-to-code-with-rust
- https://georust.org
- https://www.technologyreview.com/2023/02/14/1067869/rust-worlds-fastest-growing-programming-language

THANK YOU