Why learn Rust

if I know a memory-safe language?

Philip Kristoffersen

Philip Kristoffersen

Work

Consultant
Developer/Architect/Lead
C#, TypeScript, Java

Personal

Open Source Developer
Rust enthusiast
Rust!

BOiR
Play all your games from steam

Show you **why** and **when** you should **learn Rust**,
even if you know a memory-safe language

Goal of this talk

This talk is not

A Rust tutorial

About memory safety

About the borrow checker

If you are already a Rust pro

Stick Around

You might learn some tips
for introducing Rust

Comparing

Agenda

1 **Syntax**
What does it look like?

2 **Objectives**
What does it want to solve?

3 **Reliability**
How does it handle errors?

4 **Performance**
How fast is it?

5 **Portability**
Where can it run?

6 **Productivity**
How fast can I provide value with it?

What does Rust look like?

1. Syntax

Hello World

```rust
fn main() {
    println!("Hello, World!");
}
```

```dotnet
using System;

class Program
{
    static void Main()
    {
        Console.WriteLine("Hello, World!");
    }
}
```

```ts
console.log('Hello, World!');
```

Fibonacci

```rust
pub fn fibonacci(n:u32) -> u32 {
    if n == 0 {
        return 1;
    }
    if n == 1{
        return 1;
    }
    fibonacci(n-1) + fibonacci(n-2)
}
```

```dotnet
public static uint Fibonacci(uint n)
{
    if (n == 0)
    {
        return 1;
    }
    if (n == 1)
    {
        return 1;
    }
    return Fibonacci(n - 1) + Fibonacci(n - 2);
}
```

```ts
function fibonacci(n: number): number {
    if (n === 0) {
        return 1;
    }
    if (n === 1) {
        return 1;
    }
    return fibonacci(n - 1) + fibonacci(n - 2);
}
```

HttpGet

```rust
#[get("/hello")]
fn hello() -> &'static str {
    "Hello, World!"
}
```

Rocket

```dotnet
[HttpGet("hello")]
public IActionResult Hello()
{
    return Ok("Hello, World!");
}
```

ASP.NET

```ts
app.get('/hello', (req, res) => {
    res.send('Hello, World!');
});
```

Express

Syntax Takeaways

Familiar

Looks a lot like other languages

Unfamiliar

Adds its own syntax for its unique features like &’ and #[]

Procedural

Is not object oriented or functional but can feel like both

What does Rust try to solve?

2. Objectives

https://dotnet.microsoft.com/en-us/platform/why-choose-dotnet

https://www.typescriptlang.org/

https://www.rust-lang.org/

Productivity

Portability

Performance

.NET

Portability

Reliability

Scalability

aka. Productivity

Performance

Reliability

Productivity

Same objectives, different priorities

Pick the right tool for the job

Project

Requirements

Team

Productivity

Personal

Preference

Objective Takeaways

Rust Objectives

Performance,
Reliability, Productivity
In that order

The right tool

Find a language that
matches the situation

How does Rust handle the “unexpected”?

3. Reliability

Reading a file .NET

How many ways can this fail?

```
public static int ReadNumberFromFile(string filePath){
  var fileContent = File.ReadAllText(filePath);
  var number = int.Parse(fileContent);
  return number;
}
```

.NET

Reading a file Rust

```rust
fn read_number_from_file(file_path: &str) -> i32 {
  let file_content = fs::read_to_string(file_path);
  let number = file_content.parse::<i32>();
  return number;
}
```

```
error[E0599]: no method named `parse` found for enum `Result` in the current scope
   --> src\main.rs:25:31
    |
25  |     let number = file_content.parse::<i32>();
    |                               ^^^^^ method not found in `Result<String, std::io::Error>`
```

Result Enum

Rust has no exceptions!

No Try/Catch/Finally

Return a Result

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

Result Match

```rust
fn read_number_from_file(file_path: &str) -> i32 {
    let content = match fs::read_to_string(file_path) {
        Ok(file_contet) => file_contet,
        Err(_err) =>  String::default(),
    };
    let number = match content.parse::<i32>(){
        Ok(value) => value,
        Err(_err) => 0,
    };
    number
}
```

Easier to read Result

```rust
fn read_number_from_file(file_path: &str) -> Result<i32, ParseError> {
    let file_content = fs::read_to_string(file_path)?;
    let number = file_content.trim().parse::<i32>()?;
    Ok(number)
}

fn read_number_from_file_crash(file_path: &str) -> i32 {
    let file_content = fs::read_to_string(file_path).unwrap();
    file_content.parse::<i32>().unwrap()
}

fn read_number_from_file_default(file_path: &str) -> i32 {
    let file_content = fs::read_to_string(file_path).unwrap_or_default();
    file_content.parse::<i32>().unwrap_or_default()
}
```

Reliability Takeaways

No Runtime Surprises

The compiler will tell you if something can fail

Pedantic

You must consider how you want to handle all errors

Uncrashable

You can make code that can not crash

How fast is Rust?

4. Performance

Performance

As performant as C or C++

Faster than garbage collected languages*

Compiled, faster than interpreted languages*

* most of the time

https://programming-language-benchmarks.vercel.app/rust-vs-csharp

mandelbrot

Input: 5000

lang	code	time	stddev	peak-mem
rust	9.rs	415ms	1.0ms	4.9MB
rust	8.rs	492ms	0.5ms	4.8MB
csharp	2.cs	729ms	9.0ms	94.6MB
csharp	3.cs	3637ms	74ms	39.1MB
csharp	1.cs	4007ms	11ms	34.6MB

fasta

Input: 2500000

lang	code	time	stddev	peak-mem
rust	5c-m.rs	186ms	4.9ms	1.7MB
rust	5-m.rs	211ms	9.0ms	1.7MB
rust	1c.rs	228ms	1.4ms	1.0MB
rust	1.rs	314ms	0.3ms	1.9MB
typescript	1.ts	2113ms	52ms	47.0MB

Worth it to use Rust just for the performance?

Hardware is cheap, Developers are expensive

Zero Cost Abstractions

Which one is faster?

```
function sumWithReduce(numbers: number[]): number {
    return numbers.reduce((accumulator, currentValue) => accumulator + currentValue, 0);
}

function sumWithForLoop(numbers: number[]): number {
    let sum = 0;
    for (let i = 0; i < numbers.length; i++) {
        sum += numbers[i];
    }
    return sum;
}
```

Zero Cost Abstractions

They are the same thing

```rust
fn sum_with_for_loop(numbers: &[i32]) -> i32 {
    let mut sum = 0;
    for number in numbers {
        sum += number;
    }
    sum
}

fn sum_with_iter_sum(numbers: &[i32]) -> i32 {
    numbers.iter().sum()
}
```

Performance Takeaways

Fast

Blazingly Fast!

Zero-Cost Abstractions

The compiler optimizes away abstractions

Focus on the solution

You don't have to think as much about performance

Cheap

Very low system requirements

Where can I run Rust?

5. Portability

Portability

Command Line

WebAssembly

Networking

Embedded

https://www.rust-lang.org/

Portability Takeaways

Easy to distribute
Single statically linked executable

No runtime
Nothing to install

Embedded
Runs without a OS

Webassembly
Runs in the browser

How fast can I deliver value with Rust?

6. Productivity

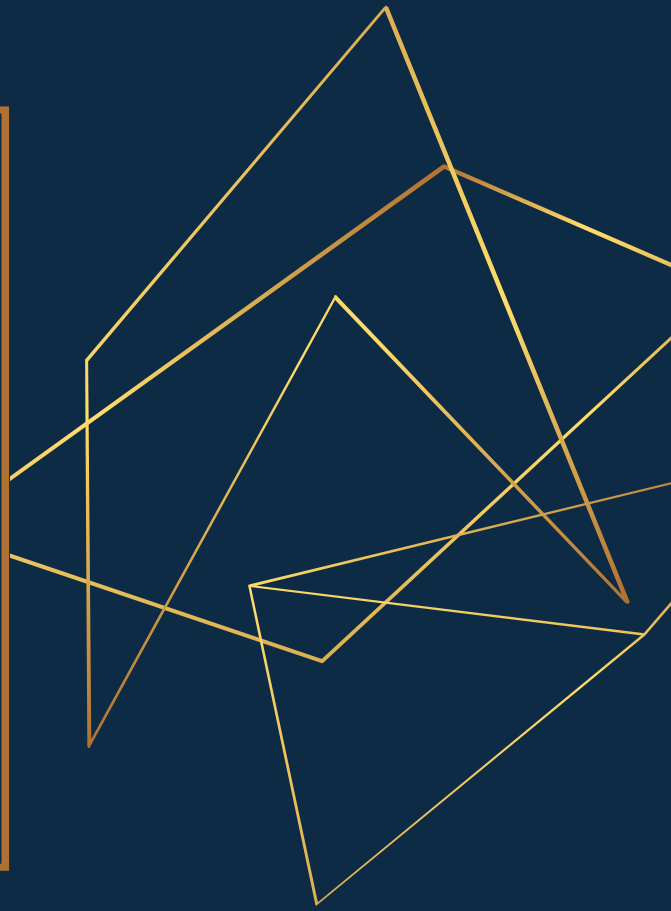
!

Opinion Time

Productivity is very opinionated

Strong static typing is for people with weak minds, and I have a weak mind

Phil Wadler



My productivity in Rust

Pros

Compiler is pedantic
No surprises ones it compiles

Borrow Checker
No memory surprises at runtime

Strong Type System
High level abstract code, with low level performance

Large Ecosystem
Everything you need at crates.io

Cons

Compiler is pedantic
A bit annoying when I just want it to run

Borrow Checker
Have to think about memory

Slow Compile
Slow compared to other languages

Productivity Takeaways

It takes a bit longer to write Rust

But you don’t have to debug as much

Why learn Rust if I know a memory-safe language?

Learn Rust when you need

Performance

Rust is very fast and uses very few resources

Reliability

Once Rust compiles it runs very reliably

Portability

You need to run code in a place other languages can't

Productivity

You want to spend less time debugging

Learn Rust if
it makes you happy

Thanks & Questions

Contact

PhilipKristoffersen@gmail.com

github.com/PhilipK

Credits

Slides theme adapted from Slidesgo template

Talk adapted from video for
https://www.youtube.com/@kodedyret