Rust

Historia de Rust

Rust es un lenguaje de programación compilado, de propósito general y multiparadigma (funcional, por procedimientos, imperativa y orientada a objetos). Su desarrollo comenzó en 2006 como un proyecto personal de Graydon Hoare, un empleado de Mozilla, y la empresa al ver el potencial de este nuevo lenguaje comenzó a patrocinarlo en 2009, antes de revelarlo al mundo oficialmente en 2010. Finalmente, su primera versión estable 1.0 fue publicada el 15 de mayo del año 2015, desarrollado por Mozilla.

El 8 de Febrero de 2021 se anunció la creación formal de la fundación Rust fundada por AWS, Huawei, Google, Microsoft y Mozilla cuyo objetivo es mantener el proyecto y apoyarlo en áreas como las patentes del proyecto o la infraestructura necesaria.

El objetivo al crear Rust fue facilitar la escritura de código, apuntando a ser similar al tiempo requerido en C++ o incluso mejor (lo que se puede ver en su sintaxis que es muy similar), y a su vez eliminar los problemas de memoria.

Características principales del lenguaje:

- Es un lenguaje compilado
- Es un lenguaje multiparadigma
- Sintaxis parecida a c/c++
- Inferencia de tipos
- Inmutabilidad por defecto para cualquier variable que declaremos
- Todo código comienza a ejecutarse por una función main
- No utiliza un garbage collector
- Es de código abierto

Usos de Rust

- Aplicaciones en línea de comandos
- Creación de modulos webassembly
- Creación de servidores
- Programación de dispositivos integrados

BNF Rust

(condición)

<if_statement> ::= "if" <expression> <block> ["else" <block>]

(bucles)

<while_loop> ::= "while" <expression> <block>

<for_loop> ::= "for" <identifier> "in" <expression> <block>

(una función)

<function> ::= "fn" <identifier> <function_signature> <block>

<function_signature> ::= "(" [<function_params>] ")" ["->" <return_type>] [<where_clause>]

<function_params>::= <identifier> ":" <type> ("," <identifier> ":" <type>)*

<return_type> ::= <type>

JavaScript

Historia de javaScript

JavaScript es un lenguaje lenguaje de programación interpretado, basada en prototipos, multiparadigma (orientada a objetos, imperativa y declarativa), de un solo hilo y dinámico. Su primera versión se lanzó en septiembre de 1995 con Netscape Navigator 2.0 bajo el nombre "LiveScript". Sin embargo, en diciembre del mismo año, Netscape se asoció con Sun Microsystems (los creadores de Java) y cambió el nombre del lenguaje a "JavaScript".

Debido a la necesidad de establecer un estándar para el lenguaje, Netscape presentó JavaScript a Ecma International en noviembre de 1996. Ecma International formó un comité técnico, TC39, para estandarizar el lenguaje. El estándar resultante se llamó ECMAScript, y la primera edición se publicó en junio de 1997.

Su objetivo y uso más extendido en la actualidad es como un lenguaje de scripting para páginas web, permitiendo que se hagan modificaciones a una página web dinámicamente del lado del cliente sin la necesidad de la interacción con el servidor, lo que fue extremadamente importante cuando el lenguaje se empezó a usar y dio paso a el desarrollo web como lo conocemos hoy en día. Además, JavaScript se puede usar para otros propósitos, como en el backend gracias a Node.JS.

Características de javaScript

- Lenguaje de programación interpretado
- Dinámicamente tipado
- Multiplataforma
- Orientado a objetos
- Manejo del DOM

Usos de JavaScript

- Crear scripts para páginas web
- Desarrollo backend (Node.JS)
- Desarrollo con aplicaciones Adobe Acrobat
- Estándar de intercambio de datos (JSON)

BNF JS

(condicional)

< statement_If>::= "if" <identifier> "(" [<parameter_list>] ")" "("<statement_list>")"

(bucles)

< statement_for>::= "for" "(" (<variable_declaration> | <statement> | null) ";"<Expression> ";" (<Expression> | null) ")"<statement>

<Statement_While> ::= "while" "(" <Expression>")"<statement>

<Expression> ::= <AssignmentExpression>

<AssignmentExpression> ::= <LeftHandSideExpression> <AssignmentOperator> <AssignmentExpression> | <ConditionalExpression>

(una función)

<function_declaration> ::= "function" <identifier> "(" [<parameter_list>] ")" <block>

<parameter_list> ::= <identifier> | <identifier> "," <parameter_list>

Ejemplo de código y tiempo de ejecución

Para comparar la velocidad de ejecución entre Rust y JavaScript, podemos usar el ejemplo de calcular los números primos hasta cierto límite. La generación de números primos es una tarea que involucra una cantidad significativa de cálculos y bucles, lo que nos permitirá ver las diferencias en el rendimiento entre ambos lenguajes.

El código en rust:

```rust
use std::time::{Instant};
fn is_prime(num: u64) -> bool {
    if num <= 1 {
        return false;
    }
    for i in 2..((num as f64).sqrt() as u64 + 1) {
        if num % i == 0 {
            return false;
        }
    }
    true
}

fn main() {      You, 8 minutes ago • first commit …
    // Obtén el tiempo de inicio
    let start_time = Instant::now();
    // Algo que queremos medir
    let limit = 100000; // Número límite hasta el cual se generarán los primos
    let mut primes = Vec::new();

    for num in 2..limit {
        if is_prime(num) {
            primes.push(num);
        }
    }
    println!("Tiempo de ejecución: {:.2?}", start_time.elapsed());
    println!("Primos encontrados: {:?}", primes);
}
```

El resultado de su ejecución:

```
warning: crate `exampleNumbers` should have a snake case name
  |
  = help: convert the identifier to snake case: `example_numbers`
  = note: `#[warn(non_snake_case)]` on by default

warning: `exampleNumbers` (bin "exampleNumbers") generated 1 warning
    Finished release [optimized] target(s) in 0.47s
PS C:\Users\PC1\Documents\Facultad\SSL\rust\jsvsRust\exampleNumbers\target\release> ./exampleNumbers.exe
Tiempo de ejecución: 15.48ms
Primos encontrados: [2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 8
 167, 173, 179, 181, 191, 193, 197, 199, 211, 223, 227, 229, 233, 239, 241, 251, 257, 263, 269, 271, 277,
379, 383, 389, 397, 401, 409, 419, 421, 431, 433, 439, 443, 449, 457, 461, 463, 467, 479, 487, 491, 499, 5
```

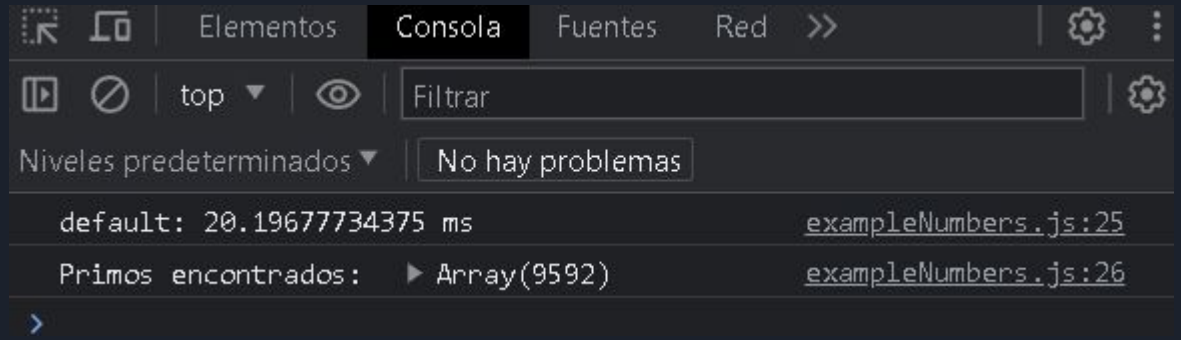
Son aproximadamente 0.01548 segundos

El mismo programa en Js

```
function isPrime(num) {
    if (num <= 1) {
        return false;
    }
    for (let i = 2; i <= Math.sqrt(num); i++) {
        if (num % i === 0) {
            return false;
        }
    }
    return true;
}

function findPrimes(limit) {
    const primes = [];
    for (let num = 2; num < limit; num++) {
        if (isPrime(num)) {
            primes.push(num);
        }
    }
    return primes;
}
console.time();
const limit = 100000; // Número límite hasta el cual se generarán los primos
const primes = findPrimes(limit);
console.timeEnd();
console.log("Primos encontrados: ", primes);
```

El resultado de su ejecución:



Son aproximadamente 0.020196777343 segundos

Otro ejemplo de código y tiempo de ejecución

Otro ejemplo de código para verificar las diferencias de velocidad entre ambos lenguajes puede ser calcular la suma de los primeros N números naturales utilizando recursión.

El código en rust:

```rust
use std::time::{Instant};
fn sum_natural_numbers(n: u64) -> u64 {
    if n == 0 {
        return 0;
    }
    n + sum_natural_numbers(n - 1)
}

fn main() {
    let n = 10000;
    let result = sum_natural_numbers(n);
    // Obtén el tiempo de inicio
    let start_time = Instant::now();

    println!("La suma de los {} primeros números naturales es: {}", n, result);
    // Obtén el tiempo de finalización
    println!("Tiempo de ejecución: {:.2?}", start_time.elapsed());
}
```

El resultado de su ejecución:

```
PS C:\Users\PC1\Documents\Facultad\SSL\rust\jsvsRust\anotherExampleRust\anotherExample> Cargo build --release
warning: crate `anotherExample` should have a snake case name
  |
  = help: convert the identifier to snake case: `another_example`
  = note: `#[warn(non_snake_case)]` on by default

warning: `anotherExample` (bin "anotherExample") generated 1 warning
    Finished release [optimized] target(s) in 0.01s
PS C:\Users\PC1\Documents\Facultad\SSL\rust\jsvsRust\anotherExampleRust\anotherExample> ./target/release/anotherexample.exe
La suma de los 10000 primeros números naturales es: 50005000
Tiempo de ejecución: 224.40µs
```

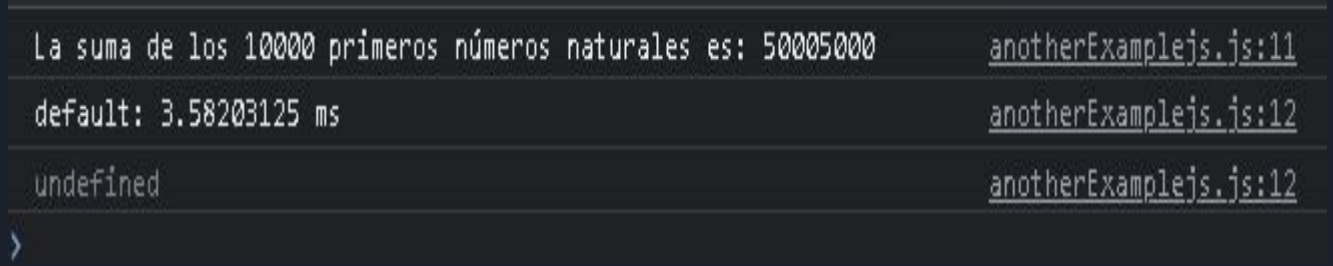
Son aproximadamente 0.0002244 segundos (`µs` es la simbología de microsegundos en rust)

El mismo programa en Js

anotherExamplejs.js > ...

```
1   console.time();
2   function sumNaturalNumbers(n) {
3       if (n === 0) {
4           return 0;
5       }
6       return n + sumNaturalNumbers(n - 1);
7   }
8
9   const n = 10000;
10  const result = sumNaturalNumbers(n);
11  console.log(`La suma de los ${n} primeros números naturales es: ${result}`);
12  console.log(console.timeEnd());
```

El resultado de su ejecución:



Son aproximadamente 0.00358203125 segundos

Conclusión

Rust resultó más rápido que Javascript en los ejemplos dados, ¿por qué?

- Rust es compilado, mientras que Javascript es interpretado.
- Rust cuenta con un sistema de macros más poderoso.
- Rust no cuenta con garbage collector

También, se debe tener en cuenta que si bien Rust suele ser más rápido que Javascript en tiempos de ejecución Javascript resulta “más sencillo” de programar, por lo que los tiempos de desarrollo suelen ser menores (aunque también cabe destacar que Rust es un lenguaje muy nuevo comparado con Javascript, por lo que contó con menos “tiempo para madurar”).

Cuestionario de la presentación

Los invitamos a escanear el QR y contestar unas preguntas referentes a la presentación: