<https://www.rust-lang.org/en-US/> rust’s eigen webpagina. Is vertrouwbaar omdat het van de taal makers zelf is. Eigenschappen rust en geleende eigenschappen.

<https://forge.rust-lang.org/platform-support.html>

platform support

<https://doc.rust-lang.org/reference/influences.html> rust’s eigen webpagina. Is vertrouwbaar omdat het van de taal makers zelf is.

“

# [Influences](https://doc.rust-lang.org/reference/influences.html" \l "influences)

Rust is not a particularly original language, with design elements coming from a wide range of sources. Some of these are listed below (including elements that have since been removed):

* SML, OCaml: algebraic data types, pattern matching, type inference, semicolon statement separation
* C++: references, RAII, smart pointers, move semantics, monomorphization, memory model
* ML Kit, Cyclone: region based memory management
* Haskell (GHC): typeclasses, type families
* Newsqueak, Alef, Limbo: channels, concurrency
* Erlang: message passing, thread failure, ~~linked thread failure~~, ~~lightweight concurrency~~
* Swift: optional bindings
* Scheme: hygienic macros
* C#: attributes
* Ruby: ~~block syntax~~
* NIL, Hermes: ~~typestate~~
* [Unicode Annex #31](http://www.unicode.org/reports/tr31/): identifier and pattern syntax

”

<https://arxiv.org/pdf/1505.07383.pdf>

“

3.1 Ownership and concurrency Because the Rust type system provides very strong guarantees about memory aliasing, Rust code is memory safe even in concurrent and multithreaded environments, but beyond that Rust also ensures data-race freedom. In concurrent programs, the data operated on by distinct threads is also itself distinct: under Rust’s ownership model, data cannot be owned by two threads at the same time. For example, the code in Figure 2 generates a static error from the compiler because after the first thread is spawned, the ownership of data has been transferred into the closure associated with that thread and is no longer available in the original thread. On the other hand, the immutable value in Figure 3 can be borrowed and shared between multiple threads as long as those threads don’t outlive the scope of the data, and even mutable values can be shared as long as they are owned by a type that preserves the invariant that mutable memory is unaliased, as with the mutex in Figure 4. Site Gecko Servo 1 thread Servo 4 threads Reddit 250 100 55 CNN 105 50 35 Table 1. Performance of Servo against Mozilla’s Gecko rendering engine on the layout portion of some common sites. Times are in milliseconds, where lower numbers are better. With relatively few simple rules, ownership in Rust enables foolproof task parallelism, but also data parallelism, by partitioning vectors and lending mutable references into properly scoped threads. Rust’s concurrency abstractions are entirely implemented in libraries, and though many advanced concurrent patterns such as work-stealing [ABP98] cannot be implemented in safe Rust, they can usually be encapsulated in a memory-safe interface.

”

<https://benchmarksgame-team.pages.debian.net/benchmarksgame/faster/rust.html>

Snelheidsverschil benchmark c++ 90% sneller keer

<https://benchmarksgame-team.pages.debian.net/benchmarksgame/program/regexredux-gcc-5.html>

met source code.

Invloed van taal op hersenen

<https://www.mpi.nl/q-a/vragen-en-antwoorden/taal-en-programmeren-in-de-hersenen>

korte uitleg rust

<https://linuxnijmegen.nl/images/pdf/PresentationRustIntroduction.pdf>

ondersteuning rust

<https://forge.rust-lang.org/platform-support.html>

rust op embeded

<https://www.slideshare.net/kellogh/glue-con14>

rust op due

<https://github.com/klangner/arduino-due-rust>

rust op due

<https://news.ycombinator.com/item?id=6268291>

rust vragen en co mabye

<https://www.reddit.com/r/rust/>

Rust Vs C++

<https://plus.google.com/+nialldouglas/posts/AXFJRSM8u2t>

Rust eigenschappen waarom wel of niet leren

<http://arthurtw.github.io/2014/12/21/rust-anti-sloppy-programming-language.html>

# IBM Why you should learn the Rust programming language

<https://www.ibm.com/developerworks/library/os-developers-know-rust/index.html>

<https://www.ibm.com/developerworks/library/os-developers-know-rust/Figure01.png>

Stackoverflow survey over talen

<https://insights.stackoverflow.com/survey/2016#technology-most-loved-dreaded-and-wanted>

`Toekomst rust ?

<https://thenewstack.io/safer-future-rust/>

KUNNEN GROTE PROJECTEN GESCHREVEN WORDEN IN RUST/ BESTAAN GROTE PRODUCTEN

<https://www.redox-os.org/>

Rust cliams waar ?

<https://dada.cs.washington.edu/research/tr/2015/03/UW-CSE-15-03-02.pdf>

functional Brian

<https://www.fpcomplete.com/blog/2018/10/is-rust-functional>

Lowlevel rust

<https://www.rust-lang.org/en-US/faq.html#low-level>

Functioneel en programeren verschil functionele talen

<https://arxiv.org/pdf/1407.5670.pdf>

rust

<https://arxiv.org/abs/1806.02693>

Rust memory menagemend talk

<https://arxiv.org/abs/1804.07608>

Klein beetje rust vernoemt onder aan(strings)

<https://arxiv.org/pdf/1711.10713.pdf>

# What can the programming language Rust do for astrophysics?

<https://arxiv.org/abs/1702.02951>

# Experience Report: Developing the Servo Web Browser Engine using Rust

<https://arxiv.org/abs/1505.07383>

wordt gesproken over concepten binnen Rust niet echt betrouw baar but still info

<https://www.reddit.com/r/rust/comments/5urar1/is_rust_likely_the_next_fastest_language_after_c/>

C++ VS RUST WHY C++ WILL NEVER DIE AND RUST (SUCKS)

<https://www.viva64.com/en/b/0324/>

C++ en rust |huidige| staat

<https://news.ycombinator.com/item?id=12744317>

Rust vs C++ review

<https://bytescout.com/blog/rust-vs-c-a-overview.html>

functioneel en rust

<https://science.raphael.poss.name/rust-for-functional-programmers.html>

rust uitleg andere school 2016

<http://users.eecs.northwestern.edu/~jesse/course/eecs395rust-wi16/>

computer

https://developer.ibm.com/articles/os-developers-know-rust/