

Rust Programmeren

Doelgroep Cursus Rust Programmeren

De cursus Rust Programmeren is bedoeld voor developers die in Rust willen leren programmeren en anderen die Rust code willen begrijpen.

Voorkennis Rust Programmeren

Ervaring met programmeren in moderne programmeertaal is wenselijk en is bevorderlijk voor een goede begripsvorming.

Uitvoering Training Rust Programmeren

De theorie wordt behandeld aan de hand van presentatie slides. De concepten worden toegelicht met demo's. De theorie wordt afgewisseld met oefeningen. De cursustijden zijn van 9.30 tot 16.30.

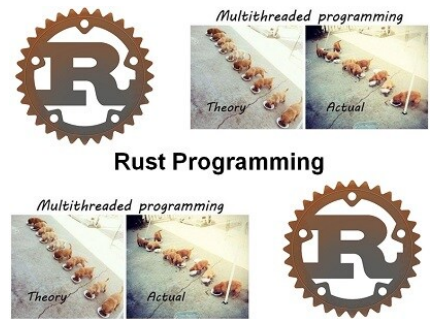
Certificering Rust Programmeren

De deelnemers krijgen na het goed doorlopen van de cursus een officieel certificaat Rust Programmeren.

Duur: 3 dagen

Prijs: € 1999

[Open Rooster](#)



Inhoud Rust Programmeren

In de cursus Rust Programmeren leren de deelnemers software te ontwikkelen met de laatste versie van de innovatieve programmeer taal Rust.

Rust Fundamentals

Rust is een nieuwe, praktische systeem programmeertaal die razendsnelle code oplevert. Rust is community driven. Met Rust voorkom je vrijwel alle crashes en data races.

Imperatief en Functioneel

Rust borduurt voort op een rijke geschiedenis van programmeer talen. Het is low-level taal met meerdere paradigma's, zowel imperatief als functioneel.

Concurrency en High Performance

Rust richt zich op veilige, high-performance, concurrent applicaties. Rust begon al voor de officiële 1.0 versie mei 2015 momentum in de industrie te krijgen, want er bestaat een duidelijke behoefte aan een nieuwe low-level systeem taal.

Traits, Borrowing en Lifetimes

In deze cursus wordt behandeld wat Rust zo uniek maakt en wordt dit toegepast op praktische problemen van systeem programmering. Onderwerpen die aan de orde zullen komen zijn : traits, generics, memory safety, move semantics, borrowing and lifetimes.

Closures en Concurrency

En ook het rijke macro-systeem van Rust, closures en concurrency komen aan de orde.

Modules Rust Programmeren

Module 1 : Rust Intro	Module 2 : Data Types	Module 3 : Flow Control
What is Rust? Rust Background Rust Momentum Rust Usage Comparisons to C Rust Applications Hello Rust Comments Formatted Printing Debug and Display Literals Operators	Primitives Tuples and Arrays Slices Custom Types Enums Constants Variable Bindings Scope Shadowing Casting Inference Alias	Expressions Flow Control if else loop Nesting and labels while for and range match Guards Binding if let while let
Module 4 : Functions	Module 5 : Modules	Module 6 : Generics
Methods Closures Capturing As Input Parameters Input Functions Type Anonymity As Output Parameters Examples from std Iterator::any Iterator::find Higher order Functions	Visibility Struct Visibility use Declaration Using super Using self File Hierarchy Crates Attributes Extern crate Dead Code Custom	Functions Implementations Parametrization over Types Traits Bounds Multiple Bounds Where Clauses Associated Items Associated Types Phantom Type Parameters Unit Clarification
Module 7 : Scoping	Module 8 : Traits	Module 9 : Standard Library
RAI Ownership and Moves Functions and Methods Mutability Borrowing and Freezing Aliasing ref Pattern Lifetimes Explicit Annotation Bounds and Coercion Static Elision	Zero cost Abstraction Traits are interfaces Derive Operator Overloading Drop Iterators Clone Designators Overload and Repeat Unsafe Operations Static dispatch Dynamic dispatch	Box, stack, heap Data Structures Vectors Strings Hashmap Threads Channels Path File I/O Pipes Wait Arguments Meta