Inhaltsübersicht

	Vorwort	xvii
	Danksagung	xix
1	Rust – Einführung	1
Teil I	Die Sprache	27
2	Syntax von Rust-Programmen	29
3	Variablen	39
4	Datentypen	43
5	Musterabgleich	59
6	Funktionen	69
7	Einführung in das Speichermodell	75
8	Generische Datentypen	97
9	Objektorientierte Konzepte	105
10	Problembehandlung in Rust	147
11	Standarddatentypen von Rust	155
12	Makros	201
13	Strukturierung von Projekten	209
14	Zusammenfassung	211

Teil 2 Fortgeschrittene Techniken 213

15 Ownership im Detail 215

16 Nebenläufige und parallele Programmierung 247

17 Testen 269

18 Webprogrammierung 297

19 Microservices 333

20 Systemnahe Programmierung 363

21 Spracherweiterungen (Language Bindings) 385

22 WebAssembly 413

23 Zusammenfassung und Ausblick 431

Inhaltsverzeichnis

Vorwort xvii

Danksagung xix

1 Rust – Einführung 1

1.1 Warum Rust? ... 1

1.1.1 Rust und der Speicher ... 1
1.1.2 Rust und Objektorientierung ... 2
1.1.3 Rust und funktionale Programmierung ... 2
1.1.4 Rust und Parallelverarbeitung ... 3

1.2 Ein Beispielprogramm ... 3

1.3 Installation von Rust ... 6

1.3.1 Installation von rustup ... 7

1.4 IDE-Integration ... 8

1.4.1 Rust Language Server und Rust-Analyzer ... 8
1.4.2 Visual Studio Code ... 9
1.4.3 IntelliJ IDEA ... 10
1.4.4 Eclipse ... 11
1.4.5 Welche Entwicklungsumgebung ist die beste? ... 12

1.5 Unsere erste praktische Erfahrung ... 13

1.6 Das Build-System von Rust ... 14

1.6.1 Die Struktur von Rust-Programmen ... 14
1.6.2 Die Erzeugung eines Packages ... 15
1.6.3 Übersetzen und Ausführen eines Packages ... 15
1.6.4 Verwaltung von Abhängigkeiten ... 17

1.6.5 Workspaces ... 22
1.6.6 Weitere nützliche Befehle von Cargo ... 24
1.7 Entwicklung der Sprache und Kompatibilität ... 25

Teil I Die Sprache **27**

2 Syntax von Rust-Programmen **29**
2.1 Programmstruktur ... 29
2.2 Anweisungsblöcke ... 30
2.3 Rangfolge von Operatoren ... 31
2.4 Gängige Kontrollflussstrukturen ... 31
2.4.1 Das If-Konstrukt ... 32
2.4.2 Das Loop-Konstrukt ... 33
2.4.3 Die While-Schleife ... 34
2.4.4 Die For-Schleife ... 35

3 Variablen **39**
3.1 Veränderbare und nicht veränderbare Variablen ... 39
3.2 Weitere Arten der Variablendefinition ... 41
3.2.1 Globale Variablen ... 41
3.2.2 Konstanten ... 41

4 Datentypen **43**
4.1 Skalare Datentypen ... 43
4.1.1 Ganzzahlen ... 43
4.1.2 Fließkommazahlen ... 46
4.1.3 Logische Werte ... 47
4.1.4 Zeichen ... 47
4.1.5 Typkonvertierung ... 48
4.2 Tupel und Felder ... 48
4.2.1 Tupel ... 49
4.2.2 Felder ... 50

4.3 Strukturierte Datentypen ... 52
4.3.1 Unterstützung bei der Initialisierung ... 53
4.4 Tupelstrukturen ... 55
4.5 Aufzählungstypen ... 56
4.5.1 In Aufzählungen eingebettete Datentypen ... 57

5 Musterabgleich 59

5.1 Das Match-Konstrukt ... 59
5.1.1 Einfache Verwendung ... 59
5.1.2 Rückgabewerte ... 60
5.1.3 Zusätzliche Bedingungen für das Muster ... 61
5.1.4 Zuweisungen im Muster ... 62
5.2 Andere Datentypen und das Match-Konstrukt ... 63
5.3 Weitere Musterabgleiche ... 65
5.3.1 Das »If Let«-Konstrukt ... 66
5.3.2 Das »While Let«-Konstrukt ... 67
5.3.3 Das Makro `matches!` ... 67

6 Funktionen 69

6.1 Referenzen auf Funktionen ... 72

7 Einführung in das Speichermodell 75

7.1 Stack und Heap ... 75
7.2 Rust und der Speicher ... 77
7.3 Das Modell für skalare Datentypen ... 79
7.3.1 Wechsel von Gültigkeitsbereichen ... 80
7.3.2 Aufruf von Funktionen ... 80
7.4 Das allgemeinere Modell ... 81
7.4.1 Wechsel von Gültigkeitsbereichen ... 84
7.4.2 Aufruf von Funktionen ... 85
7.5 Referenzen in Rust ... 86
7.5.1 Lesereferenzen auf nicht veränderbare Variablen ... 87
7.5.2 Lesereferenzen auf veränderbare Variablen ... 89
7.5.3 Veränderbare Referenzen ... 90

7.6 Verwendung von Variablen und Referenzen 94
7.7 Vor- und Nachteile des Modells 95
7.7.1 Nachteile 95
7.7.2 Vorteile 95

8 Generische Datentypen 97

8.1 Typparameter in Datenstrukturen 99
8.2 Typparameter in Funktionen 100
8.3 Typparameter in Aufzählungstypen 101

9 Objektorientierte Konzepte 105

9.1 Methoden 105
9.1.1 Die Verwendung von Typparametern 108
9.2 Module und Sichtbarkeiten 110
9.2.1 Importieren von Elementen aus anderen Namensräumen . 112
9.2.2 Hierarchische Module 114
9.2.3 Erweiterte Sichtbarkeiten 116
9.2.4 Aufteilung in mehrere Dateien 116
9.3 Traits 118
9.3.1 Erzeugung und Verwendung 120
9.3.2 Abhängigkeit von anderen Traits 122
9.3.3 Verwendung in Funktionen 123
9.3.4 Verwendung mit generischen Datentypen 124
9.3.5 Einschränkung von Typparametern mit Traits 126
9.3.6 Polymorphe Rückgabetypen 128
9.3.7 Assoziierte Datentypen 129
9.3.8 Die Größe von Instanzen 133
9.3.9 Dynamische Trait-Objekte 134
9.3.10 Traits, die Rust bereitstellt 137
9.3.11 Der Trait Drop 142
9.3.12 Das Attributsmakro Derive 143

10 Problembehandlung in Rust 147

10.1 Der Datentyp Option 147
10.2 Der Datentyp Result 149
10.3 Interoperabilität von Option und Result 151

10.4 Der ?-Operator ... 151
10.5 Nicht behebbare Fehler ... 153
10.6 Bewertung ... 154

11 Standarddatentypen von Rust 155

11.1 Kollektionen ... 155
11.1.1 Sequenzdatentypen ... 155
11.1.2 Map-Datentypen ... 160
11.1.3 Mengen ... 163
11.1.4 Verschiedene Datentypen in Kollektionen ... 165
11.1.5 Der Datentyp Slice ... 166
11.1.6 Zeichenketten ... 168
11.2 Der Datentyp Range ... 172
11.3 Closures ... 174
11.3.1 Verwendung als anonyme Funktion ... 176
11.3.2 Der umgebende Gültigkeitsbereich ... 177
11.4 Iteratoren ... 186
11.4.1 Erzeugung von Iteratoren ... 186
11.4.2 Erste Verwendung von Iteratoren ... 187
11.4.3 Weitere Verarbeitungsmöglichkeiten ... 188
11.4.4 Erzeugung von Iteratoren aus Iteratoren ... 193
11.4.5 Erzeugung neuer Kollektionen ... 198

12 Makros 201

12.1 Bekannte Makros ... 201
12.2 Beispiele für weitere Makros ... 201
12.2.1 Assertionen ... 202
12.2.2 Makros für Zeichenketten ... 203
12.3 Arten von Makros ... 204
12.4 Ein eigenes deklaratives Makro ... 205

13 Strukturierung von Projekten 209

13.1 Konfiguration des Packages ... 210

14 Zusammenfassung 211

Teil 2 Fortgeschrittene Techniken **213**

15 Ownership im Detail **215**

15.1 Näheres zum bekannten Ownership-Modell ... 216

15.1.1 Move, Copy, Clone, Borrow ... 216
15.1.2 Lifetimes ... 220
15.1.3 Die Sicherheit von Rust ... 226

15.2 Smart Pointer ... 228

15.2.1 Box ... 228
15.2.2 Rc ... 235
15.2.3 RefCell und Cell ... 239
15.2.4 Zusammenfassung ... 242

15.3 Vergleich mit anderen Sprachen ohne Ownership ... 243

16 Nebenläufige und parallele Programmierung **247**

16.1 Grundlagen ... 247

16.2 Channels ... 250

16.3 Shared State ... 251

16.3.1 Arc ... 251
16.3.2 Send und Sync ... 256
16.3.3 Mutex ... 256
16.3.4 RwLock ... 258

16.4 Einfache Parallelisierung mit Rayon ... 259

16.5 Sicherheit trotz Parallelität ... 260

16.6 Async/Await ... 261

16.7 Zusammenfassung ... 267

17 Testen **269**

17.1 Arten von Tests ... 269

17.1.1 Unit-Tests ... 270
17.1.2 Integration-Tests ... 271
17.1.3 UI-Tests ... 272
17.1.4 Testpyramide, Nachwort ... 273

17.2 Rust, Cargo und Tests ... 273

17.2.1 Platzierung von Testcode ... 273

17.3 Ausführung ... 276
17.3.1 Erwartungen der Testergebnisse (Assertions) ... 276
17.3.2 Benennung der Testfunktionen ... 278
17.4 Mocking ... 280
17.4.1 Erste Schritte ohne Framework ... 280
17.4.2 Einsatz eines Frameworks: Mockall ... 282
17.4.3 Abschließendes zu Mockall ... 291
17.5 Snapshot-Tests mit `insta` ... 291
17.6 Der Rust-Compiler sieht viel, aber nicht alles ... 293
17.6.1 Überläufe (Overflows) ... 294
17.6.2 OutOfBoundsCheck ... 294
17.6.3 Stockungen (Deadlocks) ... 295
17.7 Fazit ... 295

18 Webprogrammierung 297

18.1 Einführung ... 297
18.1.1 Warum Rust für Webprogrammierung? ... 298
18.1.2 Warum nicht Rust für Webprogrammierung? ... 300
18.1.3 Themen in diesem Kapitel ... 300
18.1.4 Eine kleine Warnung vorab ... 301
18.2 Grundlagen von Rocket ... 301
18.2.1 Handler ... 302
18.2.2 Return Types ... 304
18.2.3 Ein Blick hinter die Kulissen ... 308
18.2.4 Shared State ... 308
18.3 Das Kontaktformular ... 311
18.3.1 Routen ... 311
18.3.2 Formulare ... 312
18.3.3 Datenbankverbindung ... 313
18.3.4 Was macht Rust bis hierher so besonders? ... 317
18.3.5 Middleware ... 318
18.3.6 Guards ... 320
18.3.7 Fairings oder Guards? ... 322
18.3.8 Serverside-Templates ... 323
18.3.9 Testen mit Rocket ... 325

18.4 Betrieb ... 327
18.4.1 Logging ... 328
18.4.2 Konfiguration ... 329
18.4.3 Deployment ... 331
18.5 Fazit ... 332

19 Microservices 333

19.1 Eignet sich Rust für Microservices? ... 334
19.2 Aufteilung der Webanwendung in Microservices ... 336
19.3 Vorbereitungen ... 337
19.3.1 Build mit Docker ... 339
19.3.2 Cross Compilation ... 342
19.4 Die Microservices ... 344
19.4.1 Anfragen annehmen: der »Web«-Service ... 344
19.4.2 Gemeinsame Funktionalität ... 347
19.4.3 Speichern der Anfragen in der Datenbank ... 349
19.4.4 Mail verschicken ... 350
19.5 Betrieb ... 353
19.5.1 Metriken und Monitoring ... 353
19.5.2 Tracing ... 356
19.5.3 Skalierung ... 361
19.6 Zusammenfassung ... 361

20 Systemnahe Programmierung 363

20.1 Unsafe Rust ... 363
20.1.1 Pointer-Grundlagen ... 364
20.1.2 Unsafe in std: RefCell als Beispiel ... 365
20.2 Systemaufruf ... 366
20.2.1 Systemaufruf in Handarbeit ... 367
20.2.2 Systemaufruf mit dem Crate libc ... 368
20.3 Integration von externen Bibliotheken in Rust ... 369
20.3.1 Fallstricke ... 371

20.4 Performanceuntersuchung ... 372
20.4.1 Erste Schritte ... 373
20.4.2 Benchmarks ... 374
20.4.3 Untersuchungen ... 378
20.4.4 Optimierung ... 381
20.5 Zusammenfassung ... 384

21 Spracherweiterungen (Language Bindings) 385

21.1 Java ... 385
21.1.1 Grundsätzliches – Java ruft Rust auf ... 386
21.1.2 j4rs ... 389
21.1.3 Zusammenfassung ... 403
21.2 Node.js ... 403
21.3 Fazit ... 412

22 WebAssembly 413

22.1 Aktueller Stand von WebAssembly ... 413
22.1.1 Im Browser ... 415
22.1.2 Außerhalb des Browsers – ein Anfang ... 417
22.2 Rust & WASM ... 419
22.2.1 Warum Rust für WASM? ... 419
22.2.2 Im Browser: wasm-bindgen & wasm-pack ... 420
22.2.3 Auf dem Server ... 423
22.3 Fazit ... 430

23 Zusammenfassung und Ausblick 431

23.1 Zusammenfassung ... 431
23.2 Ausblick ... 431

Index 432