

```
fn main() {  
    println!("Rust");  
    println!("=====");  
    println!("")  
    println!("- Ownership");  
    println!("- Traits");  
    println!("- Concurrency");  
    println!("- Pattern Matching");  
}
```

```
// Rust ist eine Systemprogrammiersprache, die  
// blitzschnell läuft, Speicherfehler vermeidet  
// und Threadsicherheit garantiert.
```

- **Paradigma:** Multiparadigmen
 - (generisch, nebenläufig, funktional, imperativ, strukturiert)
- **Erscheinungsjahr:** 2010
 - erste stabile Version 2015
- **Entwickler:** Graydon Hoare (Mozilla)
- **Aktuelle Version:** 1.31 (6. Dezember 2018)
- **Typisierung:** stark, statisch, linear, Typinferenz
- **Features:**
 - Zero-Cost-Abstraktionen, Move-Semantiken
 - Garantierte Speichersicherheit, Threads ohne Data Races
 - Trait-basierte Generics, Pattern Matching, Typinferenz
 - Minimales Laufzeitsystem, Effiziente Schnittstelle zu C

```
fn main() {  
    let s = String::from("hello");  
    let len = calculate_length(s);  
    println!("The length of '{}' is {}.", s, len);  
}  
  
fn calculate_length(s: String) -> usize {  
    s.len()  
}
```

```
pub trait Summary {  
    fn summarize(&self) -> String;  
}  
  
pub struct Tweet {  
    pub username: String,  
    // ...  
}  
  
impl Summary for Tweet {  
    fn summarize(&self) -> String {  
        format!("{}", self.username)  
    }  
}
```

TODO

TODO

TODO

- intelligenter Compiler
- gutes Tooling (`cargo`, `rustfmt`)
- dünne Standard-Library (Abhängigkeit von Libraries)
- teils gewöhnungsbedürftig (Syntax, Memory-Handling)
- zwischen Rust und Go hin und her gerissen
 - Vorteile von Rust:
 - ausgeklügeltes Typsystem
 - kein Garbage Collector
 - Performance
 - Vorteile von Go:
 - bessere Standard-Library
 - einfachere Syntax
 - *noch* besseres Tooling
 - Google und Unix-Genies dahinter: Thompson, Pike, Kernighan (Buch)

Fazit: Ich werde mich weiter mit Rust beschäftigen.

TODO