```rust
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.
```

Was ist Rust?

- **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

SP1: Ownership-Konzept

```rust
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()
}
```

SP2: Traits

```rust
pub trait Summary {
    fn summarize(&self) -> String;
}

pub struct Tweet {
    pub username: String,
    //  ...
}

impl Summary for Tweet {
    fn summarize(&self) -> String {
        format!("{}", self.username)
    }
}
```

SP3: Concurrency

TODO

SP4: Pattern Matching

TODO

Technisches Team-Fazit

TODO

Persönliches Fazit - Patrick

- 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.

Persönliches Fazit - Lukas

- einige interessante Konzepte z.B. Ownership
  - kann Probleme bereiten (z.B. Stack)
- Multiplatform, Package-Manager und Build-Tool direkt eingebaut
- “intelligenter” Compiler
  - erzwingt “guten” Code
  - gibt meistens sehr gute Fehlermeldungen
- Interessante Alternative zu C
- für kleine CLI Tool sicher sehr gut geeignet
- sehr lebendige Sprache (neue Versione, Website, …)

Fazit: Weiterempfehlung erteilt