```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

Geschichte von Rust

- `2006..2010`: Privates Projekt vom Mozilla-Mitarbeiter Graydon Hoare
- `2010..2012`: Mozilla nimmt Rust unter seine Obhut
    - Firefox: 4.5M Zeilen C++
- `2012..2014`: Einbindung der Community, Weggang von Graydon Hoare
- `2014..2016`: Stabilisierung (Version 1.0.0), Fokus auf Libraries
- `2016..2018`: Produktiveinsatz (Servo, Dropbox), Redox, Version 1.31

SP1: Ownership-Konzept

```
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: Pattern Matching (Motivation)

`null` ist problematisch: nicht das Konzept, aber die Implementierung.

```
enum Option<T> {
    Some(T),
    None,
}
```

Was bringt das?

SP3: Pattern Matching (`Option<T>`)

`Option<T>` kann nicht als Wert verwendet werden. Es müssen alle *Varianten* geprüft werden:

```
match divide(&a, &b) {
    Option::Some(c) => println!("{}/{}={}", a, b, c),
    Option::None => println!("{}/{} failed", a, b),
}
```

SP3: Pattern Matching (mächtiges `switch`/`case`)

```
match guess.cmp(&secret_number) {
    Ordering::Less => println!("Too small!"),
    Ordering::Greater => println!("Too big!"),
    Ordering::Equal => println!("You win!"),
}
```

SP4: Concurrency (Thread starten)

```rust
use std::thread;

fn main() {
    let handle: thread::JoinHandle<i32> = thread::spawn(|| {
        return 42;
    });
    println!("Wait for it...");
    match handle.join() {
        Result::Ok(v) => println!("the answer: {}", v),
        Result::Err(e) => panic!("error: {:?}", e),
    }
}
```

SP4: Concurrency (Shared Sate: `Mutex`)

TODO

SP4: Concurrency (Message Passing: `Channel`)

TODO

Technisches Team-Fazit

- Ownership-Konzept: grosse Hürde, aber sinnvoll

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 (gegenüber Go):
        - ausgeklügeltes Typsystem (Generics)
        - kein Garbage Collector (Performance, Echtzeit-Anwendungen)
        - kein `null/nil`
        - «funktionaler»
    - Vorteile von Go (gegenüber Rust):
        - mächtigere Standard Library
        - schönere, einfachere Syntax
        - *noch* besseres Tooling
        - Google und Unix-Genies dahinter: Thompson, Pike, Kernighan (Buch)

Fazit: Ich beschäftige mich weiter mit Rust und Go – und ignoriere C++.

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