

Rust Basel

Rust 101 @ BaselHack 2025



About Rust-Basel

- Non-profit swiss association
- Founded in March



What we do today

1. What makes the language great (tooling, ecosystem)
2. How you do things in Rust (branches, loops, classes etc.)
3. Open Challenge: (Advent of Code 2022)



<https://github.com/yguenduez/rust-101-slides>

Goal: You can write small programs

Some Prerequisites

Install Rust

```
curl --proto '=https' --tlsv1.2 -sSf https://sh.rustup.rs | sh
```

Ecosystem - Cargo is everything

Let's create a hello world rust project

```
cargo new my_first_rust_project  
cd my_first_rust_project  
# Builds and runs the project  
cargo run  
# Just build the project  
cargo build
```

Ecosystem - project structure

your-project

```
└── src/  
    └── main.rs  
        └── (lib.rs)  
└── target/  
    └── /debug  
        └── <binary>  
└── Cargo.toml  
└── Cargo.lock
```

- **src**: Source files
- **target**: Binary files (libraries/executables)
- **Cargo.toml**: Meta info (dependencies, authors, licenses etc.)
- **Cargo.lock**: pinning dependencies to a specific version (hash)

Ecosystem - Where to find libs?

Libraries

- Crates.io
- lib.rs

Ecosystem - adding a library

```
cargo add serde -F derive  
cargo add serde_json  
cargo add --dev rstest
```

-F/-feature: libs come
with feature flags
--dev: dev/test
dependencies

Ecosystem - tooling

```
# build/runs your code  
cargo build/run (--release)  
# formats your code  
cargo fmt  
# lints your code  
cargo clippy  
# executes your tests  
cargo test  
# generates documentation  
for your crate  
cargo doc (--open)
```

```
# Updates the rust compiler  
rustup update  
# Install other toolchain  
(e.g. iOS on ARM64)  
rustup target add aarch64-  
apple-ios  
# To then build against  
that target  
cargo build --release --  
target aarch64-apple-ios
```

Second part of the workshop

<https://github.com/yguenduez/rust-101>



Basics - Core Concepts - Ownership

```
#[derive(Debug)]
struct Number(i32);

fn main(){
    let x = Number(5);
    let y = x;

    // Won't compile, as x does not own
    // Number anymore - Number got "moved"
    // to y
    println!("Number: {:?}", x);
}
```

Basics - Core Concepts - Borrowing

```
#[derive(Debug)]
struct Number(i32);

fn main(){
    let mut x = Number(5);
    let y = &x; // immutable borrow
    let z = &mut x; // mutable borrow

    // Won't compile - we cannot have a read-only
    // borrow and a mutable borrow
    // at the same time
    println!("We use y {:?} and z {:?}", y, z);
}
```

Third part of the workshop

Try to solve the first challenge of Advent-of-Code 2022, with the Rust basics you learned =).

<https://github.com/yguenduez/rust-101-aoc>



Additional Resources

Great Books for Beginners

- The Rust Programming Language
- Rust by Example

Hands-on

- Rustlings