



What is Cargo?

- ▶ Cargo is Rust's build system and package manager
- ▶ Bundled with Rust installation (`cargo --version`)
- ▶ Source files should live in *src* directory
- ▶ `crates.io` is central package registry

What is Cargo?

```
manfred@A0-PF1T704D:~/rust$ cargo new hello_world
  Created binary (application) `hello_world` package
manfred@A0-PF1T704D:~/rust$ cd hello_world/
manfred@A0-PF1T704D:~/rust/hello_world$ cargo add rand
  Updating crates.io index
  Adding rand v0.8.5 to dependencies.
    Features:
    + alloc
    + getrandom
    + libc
    + rand_chacha
    + std
    + std_rng
    - log
    - min_const_gen
    - nightly
    - packed_simd
    - serde
    - serde1
    - simd_support
    - small_rng
  Updating crates.io index
manfred@A0-PF1T704D:~/rust/hello_world$ cat Cargo.toml
[package]
name = "hello_world"
version = "0.1.0"
edition = "2021"

# See more keys and their definitions at https://doc.rust-lang.org/cargo/reference/manifest.html

[dependencies]
rand = "0.8.5"
manfred@A0-PF1T704D:~/rust/hello_world$
```

Building with Cargo

- ▶ To build a project use *cargo build* and *./target/debug/hello_cargo* to run executable
- ▶ *cargo run* can be used to build and run in one step
- ▶ For finished project use *cargo build --release*

Rust editions

- ▶ Rust language has six-week release cycle – smaller updates more frequently
- ▶ Every two or three years a new Rust edition is produced
- ▶ New editions ship as part of the six-week release process
- ▶ Editions serve different purposes for different people
- ▶ The edition key in Cargo.toml indicates which edition the compiler should use for code.
- ▶ All Rust compiler version support any edition that existed prior to that compilers release



Module system – Crates and Modules

How to organize code

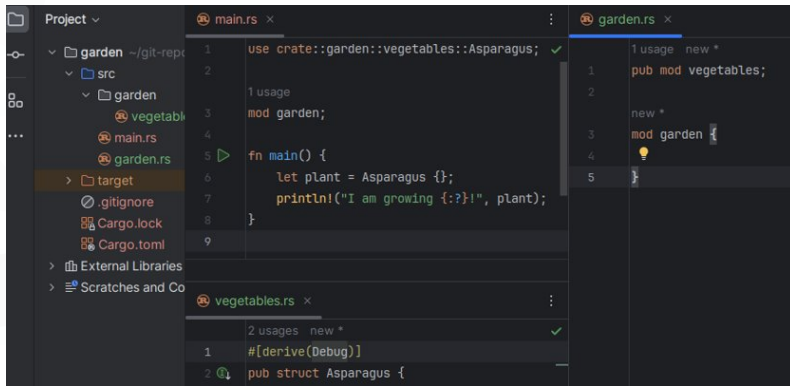
- ▶ Crate: Tree of modules producing a library or executable.
- ▶ Crate Root is source file that Rust compiler starts from to make up root module of crate
 - ◇ Binary Crates – Crate Root: `src/main.rs`
 - ◇ Library Crates – Crate Root: `src/lib.rs`
- ▶ Package: A bundle of one or more crates
- ▶ Module: Collection of items like functions, structs, traits and impl blocks
 - ◇ The `use` keyword creates shortcuts to items
- ▶ Paths: A way of naming an item to show Rust where to find an item in a module tree
 - ◇ Absolute path – Full path starting from crate root
 - ◇ Relative Path – Starts from current module (`self`, `super`)

How to organize code

When declaring a module in crate root, the compiler looks in these places

- ▶ Inline
- ▶ In `src/<module-name>.rs`
- ▶ In `src/<module-name>/mod.rs`

How to organize code



The screenshot shows an IDE with a project named 'garden' located at '~/git-repo'. The project structure is as follows:

- garden ~/git-repo
 - src
 - garden
 - vegetables.rs
 - main.rs
 - garden.rs
 - target
 - .gitignore
 - Cargo.lock
 - Cargo.toml
 - External Libraries
 - Scratches and Snippets

The code editor displays three files:

- main.rs**:

```
1 use crate::garden::vegetables::Asparagus; ✓
2
3 1 usage
4 mod garden;
5 fn main() {
6     let plant = Asparagus {};
7     println!("I am growing {:?}!", plant);
8 }
9
```
- garden.rs**:

```
1 1 usage new *
2 pub mod vegetables;
3
4 new *
5 mod garden {
```
- vegetables.rs**:

```
2 2 usages new * ✓
1 #[derive(Debug)]
2 pub struct Asparagus {
```

References

1. *module-system-title-image*
2. *cargo-title-image*