



# rust



```
# windows:
```

```
https://static.rust-lang.org/rustup/dist/x86_64-pc-windows-msvc/rustup-init.exe
```

```
msvc windows sdk
```

```
# linux:
```

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

```
# 
```

```
source "$HOME/.cargo/env"
```

## rustc --version ,

```
# 
```

```
rustc --version
```

```
# 
```

```
rustup show
```

```
# 
```

```
rustup update
```

```
# 
```

```
rustup doc
```

```
# 
```

```
rustup self uninstall
```



~~vscode, rust-analyzer~~ [rust-lang.org](https://rust-lang.org) 

## hello, world

```
# vi main.rs

fn main() {
    println!("Hello, world!");
}

#  rustc 
rustc main.rs
```

## cargo

```
# 
cargo --version

# 
cargo new hello_cargo

#  debug 
cargo build

#  release 
cargo build --release

# 
cargo run

# 
```

cargo check



```
/
/Cargo.toml          # [ ] [ ] [ ] [ ] [ ]

/src                 # [ ] [ ] [ ] [ ]
└-- /main.rs        # main
└--
└--

/target             # [ ] [ ] [ ] [ ]
└-- /debug          # debug [ ] [ ]
└-- /release        # release [ ] [ ]
```

```
[package]
name = "project_name"
version = "0.1.0"
edition = "2023"
```

```
[dependencies]
```



```
# [ ] [ ] [ ] [ ] [ ] [ ]
let x = 5;

# [ ] [ ] [ ] [ ] [ ]
let mut x = 5;
x=10;

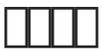
# [ ] [ ] [ ]
let x = 9;
```

```

fn function_name(){
}

// 函数
// 函数 -> 返回值
fn fun_name() -> u32 {
    // 返回值
    // 55
    // 返回 return
    // return 55;
}

```



```

struct Employ {
    name: String,
    age: u32,
    sex: String,
}

fn main() {
    let mut em: Employ = Employ {
        name: String::from("张三"),
        age: 30,
        sex: String::from("a"),
    };
    em.name = String::from("donglietao");
    println!("{}", em.name);
}

```



```

fn main() {
    let em = get_employ();
    // 函数
    let em2 = Employ { ..em };
}

```

```

println!("{}", em.name);
println!("{}", em2.name);

// 打印
println!("{}", em2);
}

fn get_employ() -> Employ {
    let mut em: Employ = Employ {
        name: String::from("张三"),
        age: 30,
        sex: String::from("a"),
    };
    em.name = String::from("donglietao");
    return em;
}

// 打印
#[derive(Debug)]
struct Employ {
    name: String,
    age: u32,
    sex: String,
}

```

打印

```

fn main() {
    let em = Employ::new(String::from("张三"), 33, String::from("1"));
    println!("{}", em.to_string());
}

// 打印
#[derive(Debug)]
struct Employ {
    name: String,
    age: u32,
    sex: String,
}

```

```
// impl
impl Employ {
    //
    fn new(name: String, age: u32, sex: String) -> Employ {
        return Employ {
            name: name,
            age: age,
            sex: sex,
        };
    }

    //
    // self
    fn to_string(self) -> String {
        return self.name;
    }
}
```

enum

```
enum Sex {
    Main = 0,
    Woman = 1,
}
```



rust 4

- package:
- crate
- modules use
- path

## package crate

`cargo new project` `crate` `src/main.rs` `src/lib.rs` (library) `package`

## module

```
/Cargo.toml
/src/
/src/main.rs
/src/ffmpeg_encoder/mod.rs
/src/ffmpeg_encoder/book.rs
/src/ffmpeg_encoder/people.rs
```

`crate` `src/main.rs` `src/lib.rs`

```
# src/main.rs

// 
use crate::ffmpeg_encoder::book::Book;
use crate::ffmpeg_encoder::people::Employ;
use crate::ffmpeg_encoder::people::Sex;
use crate::ffmpeg_encoder::Shop;

// root module
pub mod ffmpeg_encoder;
```

```

fn main() {

    println!("{}", Shop::to_string());

    println!("{}", Book::new().to_string());

    let em = Employ::new(String::from("[]"), 33, Sex::Woman);
    println!("{}", em.to_string());
}

```

```

# rust
# src/ffmpeg_encoder/mod.rs
# src/ffmpeg_encoder.rs

pub mod book;
pub mod people;

pub struct Shop {}

impl Shop {
    pub fn to_string() -> String {
        return String::from("Shop to_string");
    }
}

```

```

# src/ffmpeg_encoder/book.rs

pub struct Book;

impl Book {
    pub(crate) fn new() -> Book {
        Book {}
    }

    pub(crate) fn to_string(&self) -> String {
        return String::from("Book to_string");
    }
}

```

```
}  
}
```

```
# Person  
# src/ffmpeg_encoder/people.rs  
  
#[derive(Debug)]  
pub struct Employ {  
    name: String,  
    age: u32,  
    sex: Sex,  
}  
  
impl Employ {  
    pub fn new(name: String, age: u32, sex: Sex) -> Employ {  
        return Employ {  
            name: name,  
            age: age,  
            sex: sex,  
        };  
    }  
    // Personself  
    pub fn to_string(&self) -> String {  
        return self.name.to_string();  
    }  
}  
  
#[derive(Debug)]  
pub enum Sex {  
    Main = 0,  
    Woman = 1,  
}
```





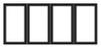
```
let json_str = json_obj.to_string();
log::info!("{}", json_str);

// 从JSON字符串中解析
let json_obj: Person = serde_json::from_value(json_str).unwrap();
log::info!("{}", json_obj.name);

return (StatusCode::OK, Json(json_obj));
}
```



- <https://lib.rs/>
- <https://crates.io/>



```
[dependencies]
# web []
axum = "0.7.2"
# []I0[]
tokio = { version = "1.35.1", features = ["full"] }
# http client
request = "0.11.23"
# []
log = "0.4.20"
# []
simple_logger = "4.3.0"
# []
serde = { version = "1.0", features = ["derive"] }
# json[]
serde_json = "1.0.68"
```

# RESTful API



```
[dependencies]
# web
axum = "0.7.2"
# I/O
tokio = { version = "1.35.1", features = ["full"] }
# http client
reqwest = "0.11.23"
# 
log = "0.4.20"
# 
simple_logger = "4.3.0"
# 
serde = { version = "1.0", features = ["derive"] }
# json
serde_json = "1.0.68"
```

## http server

```
# src/main.rs

// 
use axum::{
    http::StatusCode,
    response::{Html, IntoResponse, Json},
    routing::{get, post},
    Router,
};
use log::{debug, info, trace, warn};
use serde::{Deserialize, Serialize};
use serde_json::json;
use simple_logger::SimpleLogger;
```

```

#[tokio::main]
async fn main() {
    // 初始化
    SimpleLogger::new().init().unwrap();
    debug!("power by iovhm.com");
    warn!("©copyRight");
    trace!("2023-12-27");
    info!("application start");

    let app = Router::new()
        // GET /
        .route("/", get(root));

    let listener = tokio::net::TcpListener::bind("0.0.0.0:3000").await.unwrap();
    info!("server listen at 0.0.0.0:3000");

    axum::serve(listener, app).await.unwrap();
}

#[derive(Serialize, Deserialize)]
struct Person {
    name: String,
    age: u32,
}

async fn root() -> impl IntoResponse {
    info!("entry root");

    // 返回JSON
    let json_obj = json!({"name": "John Doe", "age": 30});

    return Json(json_obj);
}

```