kintsugi-stack-rust

> kintsugi-stack-rust

1. Getting Started

1.1. Installation

- Rust Lang: Rust Install https://doc.rust-lang.org/book/ch01-01-installation.html
  - Windows
    - Install Linux, Just kidding !!
    - https://visualstudio.microsoft.com/downloads/
      - Install VS
      - Install VSC
      - Install Build Tools for Visual Studio
        - then Restart Computer
    - https://rust-lang.org/tools/install/
      - Install Rust
    - `rustup toolchain install stable-x86_64-pc-windows-gnu`
    - `rustup default stable-x86_64-pc-windows-gnu`
  - Linux: `$ curl --proto '=https' --tlsv1.2 https://sh.rustup.rs -sSf | sh`
- Rust Server Dev: Rust Analyzer Install https://marketplace.visualstudio.com/items?itemName=rust-lang.rust-analyzer

1.2. Hello, World!

- rust code file extension `.rs`

```
// 1_1_hello_world.rs
fn main(){
    println!("Hello, World! ")
}
```

- Compile command

```
rustc main.rs
```

- Rust Binary Run command

```
./main
```

2_Cargo

- Cargo:
    - Rust's Build System
    - Package manager
    - Builtin When we Install Rust (Painpoint of other prog. lang.)
- Compile command

```
rustc main.rs
```

- Rust Binary Run command

```
./main
```

- Cargo version check

```
cargo --version
```

- Create New Cargo Project

```
cargo new hello_cargo
```

- File Organisation
    - `Cargo.toml`
        - package config file
    - `.gitignore`
        - default code ver. ignore file
        - ignore flags for git ver.
    - `\src`
        - contains actual code
        - `main.rs`
            - Starter code
- Build command
    - Build
    - Create `Cargo.lock`
    - contain dependencies
    - Create `\target`
    - contain `\debug`
        - contain our actual executable
        - other supporting stuff

```
cargo build
```

- run command

```
cargo run
```

- help command
  - to view all commands

```
cargo help
```

- check command
  - check your prog. for err.
  - without producing any executable
  - faster than running the prog.

```
cargo check
```