Two Years stable

Rust Zürichsee Meetup
Locationsponsor: Liip.ch

Organisation of the evening

- **Beginners workshop:**
  - Speaker: Sebastian
  - Topics:
    - Ownership & Borrowing
- **Advanced workshop:**
  - Speaker: Stefan
  - Topics:
    - CLI Argumentparsing with docopt
    - Async Networking with tokio
- **Cake:**

Argument Parsing

How to get from text

```
Shotgun Gameserver

Usage:
  shotgun_gameserver [--listen=<IP>] [--port=<PORT>]
  shotgun_gameserver (-h | --help)

Options:
    --port=<PORT>    The port to listen on [default: 6000]
    --listen=<IP>    The socket address to listen on [default: ::1]
```

to a rust struct?

```
#[derive(Debug,RustcDecodable)]
struct Args {
    flag_port  : u16,
    flag_listen: String,
}
```

Argument Parsing

1. Import docopt and rustc-serialize
2. Add extern crate
3. Make string static and create the struct
   (The stuff from before)
4. Add runtime-parsing:

```
[dependencies]
docopt = "^0.7"
rustc-serialize = "^0.3"
```

```
extern crate rustc_serialize;
extern crate docopt;
```

```
#[derive(Debug,RustcDecodable)]
struct Args {
    flag_port  : u16,
    flag_listen: String,
}

static USAGE: &'static str = "
Shotgun Gameserver

Usage:
  shotgun_gameserver [--listen=<IP>] [--port=<PORT>]
  shotgun_gameserver (-h | --help)

Options:
    --port=<PORT>    The port to listen on [default: 6000]
    --listen=<IP>    The socket address to listen on [default: ::1]
";
```

```
let args: Args = docopt::Docopt::new(USAGE).and_then(|d| d.decode())
                                           .unwrap_or_else(|e| e.exit());
```

Async Networking

What kind of tools do we have?

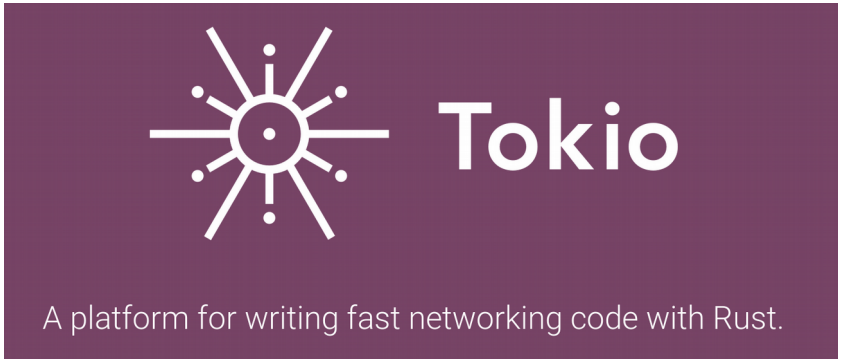
- **Sync IO with Threads using pools**
- **select**
  `man 3 select`

```
NAME
       select – synchronous I/O multiplexing

SYNOPSIS
       #include <sys/select.h>

       int select(int nfds, fd_set *restrict readfds,
           fd_set *restrict writefds, fd_set *restrict errorfds,
           struct timeval *restrict timeout);
```

- **event-loops**



Tokio Architecture

- **Futures**
- **Tokio IO – I/O Traits and Combinators**
- **Tokio Core – Byte Streams, Frames**
- **Tokio Proto – Independent Layers (eg. TLS)**
- **Tokio Service – Request/Response (eg. HTTP)**

The story of this project

- **Source Code: https://github.com/dns2utf8/shotgun**
- **Will be continued**

Questions

Cake

Thank you for your attention