DEVELOPING A FASTER AND SAFER WEB WITH RUST

Stanko Krtalic Rusendic

github.com/Stankec
@monorkin

01

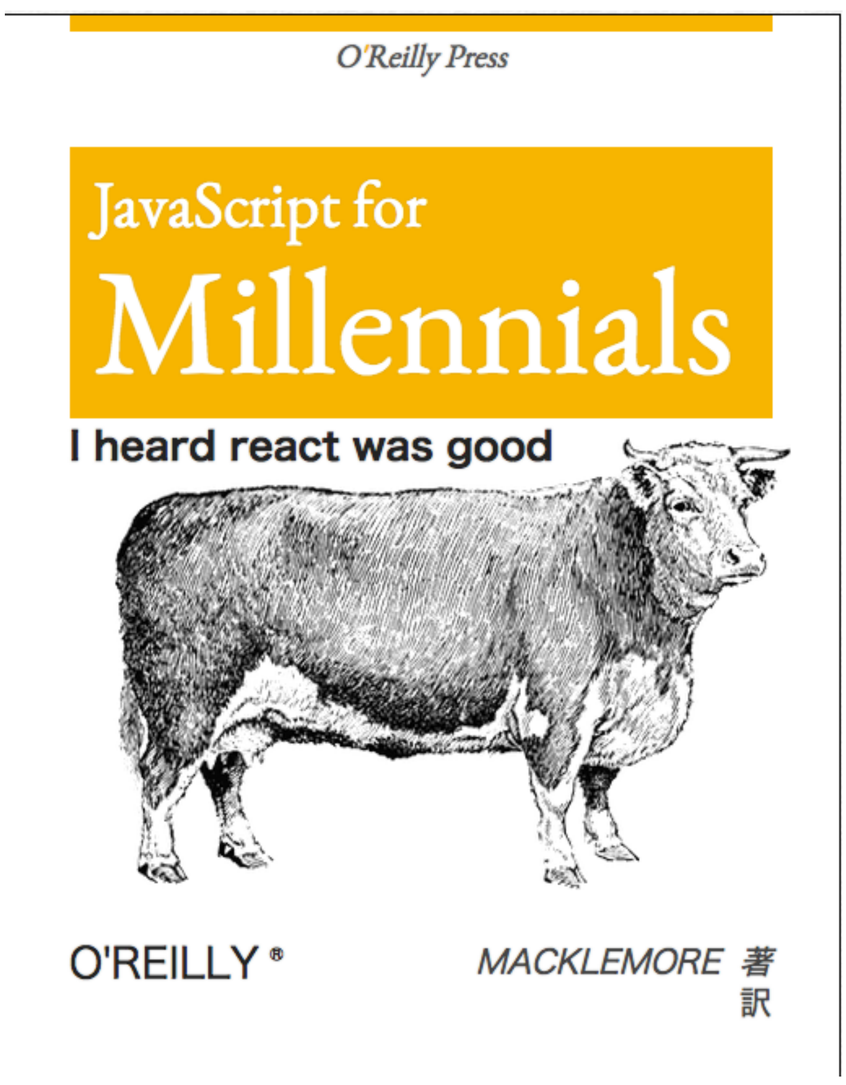
OMG RUST

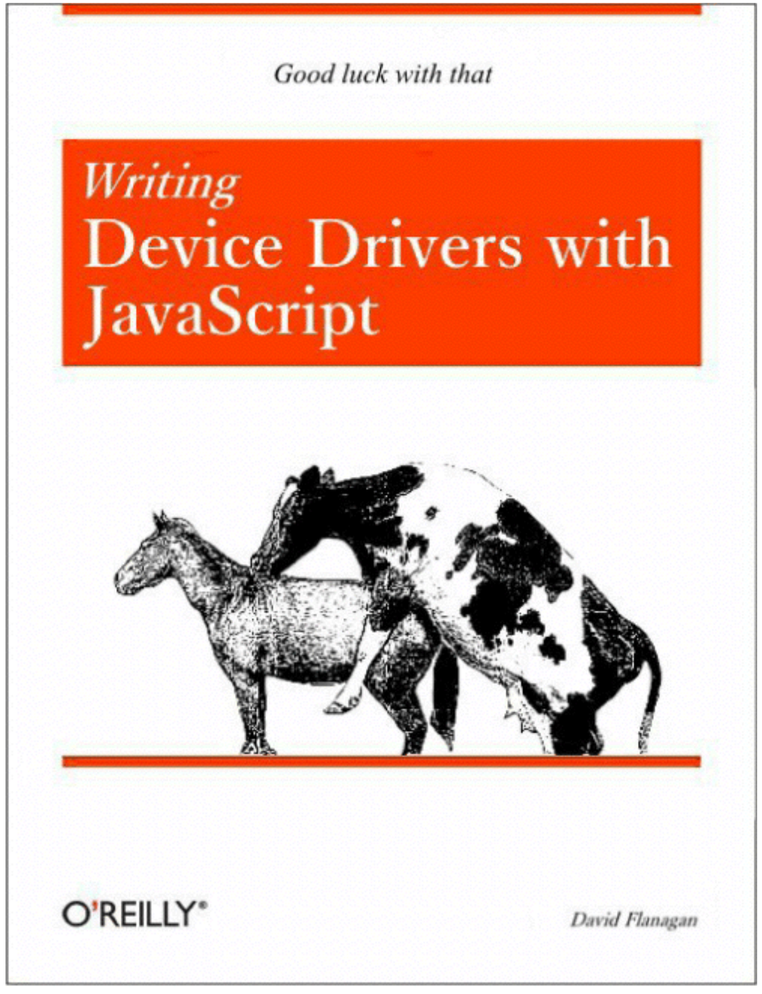
You might as well just kill yourself right now

Web Development With Assembly

O'REILLY®

Bob Johnson
with His Therapist





Low level JavaScript

4th Edition

Oh Fuck

The Acid is Kicking In

Jesus Christ the Walls are Melting

O'REILLY®

Arnold Robbins

Safety

02

AWAY

Are we *web* yet?

You can build stuff!

Rust has a mature HTTP stack and various frameworks enable you to build APIs and backend services quickly. While increasingly more databases drivers become available, ORMs and connections to external services (like search or worker queues) are still scarce. Looking farther, it doesn't necessarily get better. Though there is significant support for base needs (like data compression or logging), a lot more web-specific needs are still unmet and immature.

Can I replace my Rails/Django/Flask already?

HTTP
Crypto
Database
Email
Serialization
Logging

HTTP

~~Crypto~~

Database

~~Email~~

Serialization

~~Logging~~

03

HTTP

Iron

Rocket

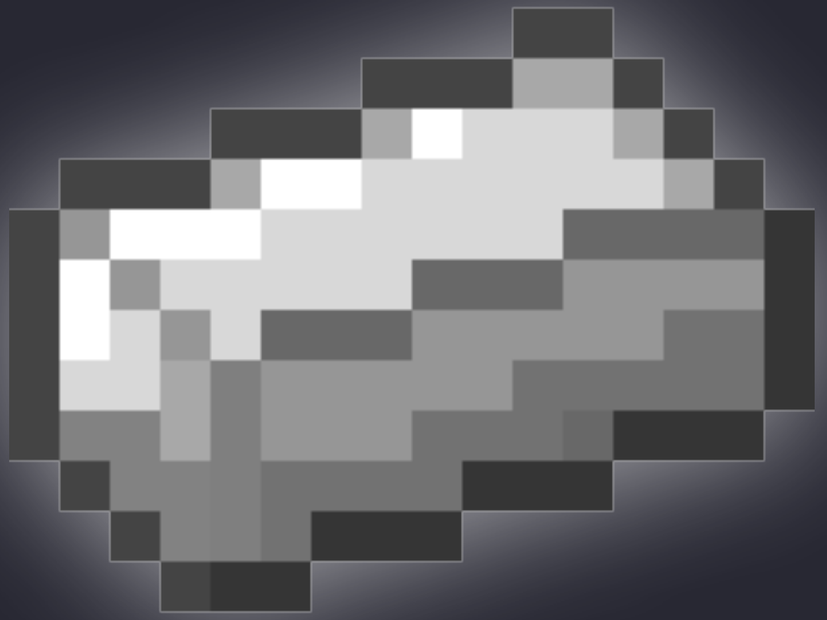
```rust
fn main() {
    // Create a router to specify which endpoint coresponds to which method
    let mut router = Router::new();
    router.get("/", index, "landing");
    //          ^^^  ^^^^^  ^^^^^^^^^^
    //          path method name

    // Create a mountpoint for the application
    let mut mount = Mount::new();
    // mount the router at the root path
    mount.mount("/", router);

    // Create a request / response chain
    let mut chain = Chain::new(mount);

    let server = Iron::new(chain).http("127.0.0.1:3000");
}

fn index(request: &mut Request) -> IronResult<Response> {
    Ok(Response::with((status::Ok, None, "Hello World!")));
}
```

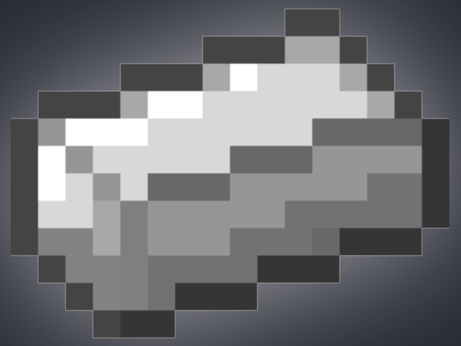


```rust
fn main() {
    rocket::ignite()
        .mount("/", routes![index])
        .launch()
}

#[get("/")]
fn index() -> String {
    "Hello World!".to_string()
}
```

Rocket

758,044 req/sec



579,227 req/sec

```rust
#[get("/hello/<name>/<age>")]
fn hello(name: &str, age: u8) -> String {
    format!("Hello, {} year old named {}!", age, name)
}

struct UserLogin {
    username: String,
    password: String
}

#[post("/login", data = "<user_form>")]
fn login(user_form: Form<UserLogin>) -> String {
    format!("Hey! {} your password has been stolen!", user_form.username);
}

struct Message {
   contents: String
}

#[put("/<id>", data = "<message>")]
fn update(id: ID, message: JSON<Message>) -> JSON<Value> {
    JSON(json!{ "status": "ok", "message": message.contents })
}
```

```rust
#[get("/user/<id>")]
fn user(id: usize) -> String {
    "You sent the ID as an usize".to_string()
}

#[get("/user/<id>", rank = 2)]
fn user_int(id: isize) -> String {
    "You sent the ID as an isize".to_string()
}

#[get("/user/<id>", rank = 3)]
fn user_str(id: &str) -> String {
    "You sent the ID as a string".to_string()
}
```

```rust
struct APIKey(String);

/// Returns true if `key` is a valid API key string.
fn is_valid(key: &str) -> bool {
    key == "valid_api_key"
}

impl<'a, 'r> FromRequest<'a, 'r> for APIKey {
    type Error = ();

    fn from_request(request: &'a Request<'r>) -> request::Outcome<APIKey, ()> {
        let keys: Vec<_> = request.headers().get("x-api-key").collect();
        if keys.len() != 1 {
            return Outcome::Failure((Status::BadRequest, ()));
        }

        let key = keys[0];
        if !is_valid(keys[0]) {
            return Outcome::Forward(());
        }

        return Outcome::Success(APIKey(key.to_string()));
    }
}
```

04

DATABASE

DIESEL

migrations/20170517000l/up.sql

```
CREATE TABLE posts (
  id SERIAL PRIMARY KEY,
  title VARCHAR NOT NULL,
  body TEXT NOT NULL,
  published BOOLEAN NOT NULL DEFAULT 'f'
)
```

migrations/20170517000l/down.sql

```
DROP TABLE posts
```

src/databse.rs

```rust
#[macro_use] extern crate diesel_codegen;

pub mod schema;
pub mod models;
```

src/models.rs

```rust
#[derive(Queryable)]
pub struct Post {
    pub id: i32,
    pub title: String,
    pub body: String,
    pub published: bool,
}
```

src/schema.rs

```rust
infer_schema!("dotenv:DATABASE_URL");
```

```rust
fn main() {
    use diesel_demo::schema::posts::dsl::*;

    let connection = establish_connection();
    let results = posts.filter(published.eq(true))
        .limit(5)
        .load::<Post>(&connection)
        .expect("Error loading posts");

    println!("Displaying {} posts", results.len());
    for post in results {
        println!("{}", post.title);
        println!("----------\n");
        println!("{}", post.body);
    }
}
```

```rust
fn main() {
    let database_connection = establish_connection();

    rocket::ignite()
        .manage(database_connection)
        .mount("/", routes![index])
        .launch()
}

#[get("/")]
fn index(database_connection: State<PgConnection>) -> String {
    let post = posts.filter(published.eq(true))
        .first()
        .load::<Post>(&database_connection)
        .expect("Error loading posts");

    post.title.to_string()
}
```

```rust
#[get("/count")]
fn count(hit_count: State<HitCount>) -> String {
    let current_count = hit_count.0.load(Ordering::Relaxed);
    format!("Number of visits: {}", current_count)
}

fn main() {
    rocket::ignite()
        .manage(Config::from(user_input))
        .launch()
}
```

05

SERIALIZATION

```rust
#[derive(Serialize, Deserialize, Debug)]
struct Point {
    x: i32,
    y: i32,
}

fn main() {
    let point = Point { x: 1, y: 2 };

    // Convert the Point to a JSON string.
    let serialized = serde_json::to_string(&point).unwrap();

    // Prints serialized = {"x":1,"y":2}
    println!("serialized = {}", serialized);

    // Convert the JSON string back to a Point.
    let deserialized: Point = serde_json::from_str(&serialized).unwrap();

    // Prints deserialized = Point { x: 1, y: 2 }
    println!("deserialized = {:?}", deserialized);
}
```

serde_json crates.io v1.0.2 build passing build passing — 958,601
A JSON serialization file format
Documentation Repository

serde_yaml crates.io v0.7.0 — 48,873
YAML support for Serde
Documentation Repository

serde_bytes crates.io v0.10.0 — 1,562
Optimized handling of `&[u8]` and `Vec<u8>` for Serde
Homepage Documentation Repository

serde_derive_internals crates.io v0.15.0 build passing — 25,849
AST representation used by Serde derive macros. Unstable.
Homepage Documentation Repository

serde_codegen_internals crates.io v0.14.2 build passing — 478,539
AST representation used by Serde codegen. Unstable.
Homepage Documentation Repository

serde_derive crates.io v1.0.5 build passing — 307,429
Macros 1.1 implementation of #[derive(Serialize, Deserialize)]
Homepage Documentation Repository

serde_test crates.io v1.0.5 build passing — 27,675
Token De/Serializer for testing De/Serialize implementations
Homepage Documentation Repository

serde_osc crates.io v0.4.1 — 104
Serialization and deserialization of Open Sound Control (OSC) packets using serde
Documentation Repository

serde-hjson crates.io v0.8.1 — 3,556
Hjson serialization file format
Repository

```rust
#[get("/")]
fn index(database_connection: State<PgConnection>) -> JSON<Value> {
    let post = posts.filter(published.eq(true))
        .first()
        .load::<Post>(&database_connection)
        .expect("Error loading posts");

    JSON(
        json!({
            "post": {
                "title": post.title.to_string()
            }
        })
    )
}
```

06

ECOSYSTEM

Questions?