درس ۸

Rust

Result و ?

```rust
// main.rs
let text =
  fs::read_to_string(p)?;
Ok(text.len())
```

یک کورس رایگان از MyWebUniversity

برنامه‌نویسی Rust

درس ۸

مدیریت خطا با Result

panic!، Result، عملگر ?، خطاهای سفارشی، thiserror و anyhow



panic!: خطاهای غیرقابل جبران

panic! پروگرام را با یک پیام متوقف می‌کند. برای باگ‌ها است، موقعیت‌هایی که هرگز نباید رخ دهند.

unwrap و expect روی None یا Err، panic می‌کنند. در مثال‌های سریع و آزمایش‌ها خوب هستند، اما ...

terminal

```
$ cargo run
thread 'main' panicked at src/main.rs:4:37:
config.toml must exist: Os { code: 2, kind: NotFound }
note: run with `RUST_BACKTRACE=1` to display a backtrace
```

01

Result

تابع‌هایی که ممکن است ناکام شوند Result برمی‌گردانند: Ok با مقدار، یا Err با یک خطا. مثل ...

روی نتیجه match کنید، یا از method هایی مثل unwrap_or_else و map_err استفاده کنید.

main.rs

```rust
use std::fs;

match fs::read_to_string("scores.txt") {
    Ok(text) => println!("{} lines", text.lines().count()),
    Err(e) => eprintln!("Could not read file: {e}"),
}
let n: i32 = "42".parse().unwrap_or(0);
```

02

عملگر ؟

علامت سوال بعد از یک Result، اگر خطایی باشد آن را زود برمی‌گرداند، یا اگر Ok باشد مقدار را ...

زنجیره‌های طولانی match را به کود پاک و کوتاه تبدیل می‌کند. main هم می‌تواند Result برگرداند.

main.rs

```rust
fn average_from_file(path: &str) -> Result<f64, Box<dyn std::error::Error>> {
    let text = std::fs::read_to_string(path)?;
    let nums: Vec<f64> = text.lines()
        .map(|l| l.trim().parse::<f64>())
        .collect::<Result<_, _>>()?;
    Ok(nums.iter().sum::<f64>() / nums.len() as f64)
}
```

03

نوع‌های خطای سفارشی

برای کتابخانه‌ها، یک enum خطا با یک حالت برای هر ناکامی تعریف کنید. crate thiserror کود ...

تبدیل‌های From اجازه می‌دهند علامت سوال خطاهای سطح پایین‌تر را خودکار تبدیل کند.

error.rs

```rust
#[derive(Debug, thiserror::Error)]
enum GradeError {
    #[error("cannot read file: {0}")]
    Io(#[from] std::io::Error),
    #[error("bad score on line {line}")]
    BadScore { line: usize },
}

fn load(p: &str) -> Result<Vec<f64>, GradeError> { /* ... ? ... */ }
```

04

anyhow برای اپلیکیشن‌ها

در اپلیکیشن‌ها، crate anyhow یک نوع خطای آسان می‌دهد. context به هر خطا هنگام بالا رفتن یک ...

قاعدهٔ ساده: thiserror برای کتابخانه‌ها، anyhow برای اپ‌ها.

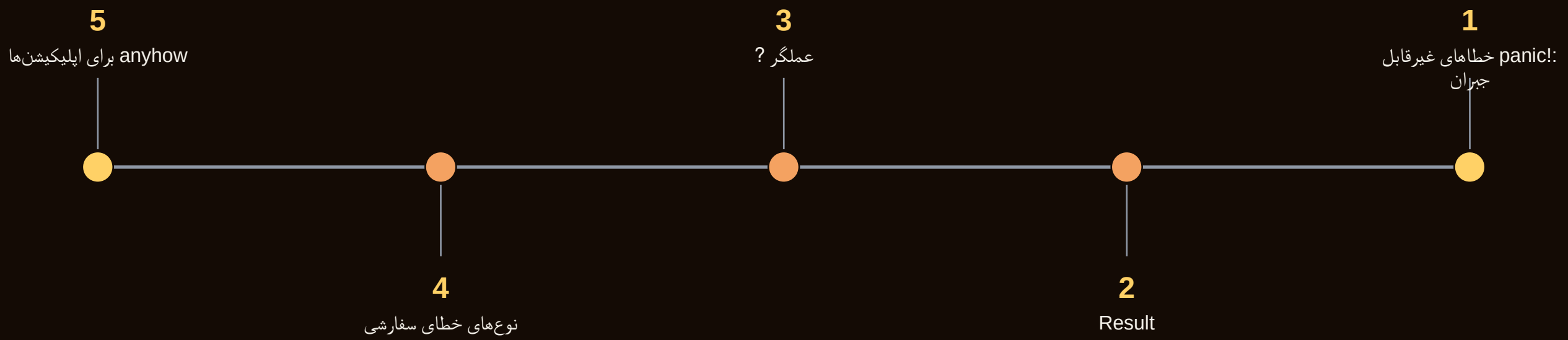
main.rs

```rust
use anyhow::{Context, Result};

fn main() -> Result<()> {
    let cfg = std::fs::read_to_string("config.toml")
        .context("reading config.toml")?;
    let port: u16 = cfg.trim().parse().context("port must be a number")?;
    println!("port {port}");
    Ok(())
}
```

05

نقشهٔ درس


1
panic!: خطاهای غیرقابل جبران
2
Result
3
عملگر ?
4
نوع‌های خطای سفارشی
5
anyhow برای اپلیکیشن‌ها

منابع و مطالعهٔ بیشتر

terminal

```
$ cargo run
thread 'main' panicked at src/main.rs:4:37:
config.toml must exist: Os { code: 2, kind: NotFound }
note: run with `RUST_BACKTRACE=1` to display a backtrace
```

RUST PROGRAMMING · 8

main.rs

```rust
fn average_from_file(path: &str) -> Result<f64, Box<dyn std::error::Error>> {
    let text = std::fs::read_to_string(path)?;
    let nums: Vec<f64> = text.lines()
        .map(|l| l.trim().parse::<f64>())
        .collect::<Result<_, _>>()?;
    Ok(nums.iter().sum::<f64>() / nums.len() as f64)
}
```

تشکر از تماشای شما

MyWebUniversity.com

آموزش رایگان تکنالوژی از سال ۲۰۰۰ — انگلیسی و دری

@MyWebUniversity-Dari · @MyWebUniversity