

Error Management



Michael VanSickle

@vansimke



Introduction



Errors in Go

Error Mechanics

Errors vs Panics



Things Don't Always Go to Plan



Open a file that isn't there



Request data from a web service, but it doesn't respond



Divide a number by zero



Run out of memory



Access an uninitialized pointer



Are these results surprising or unexpected?

Not when running at scale!



Errors are values.

Go Proverbs - <https://go-proverbs.github.io/>



Errors in Go

```
f, err := os.Open("path/to/file")
if err != nil {
    // handle the error
}
defer f.Close()
```

- **consider error handling immediately**
- **simplify code review and understanding**
- **improve production stability**



Don't just check errors, handle them gracefully.

Go Proverbs - <https://go-proverbs.github.io/>



Demo



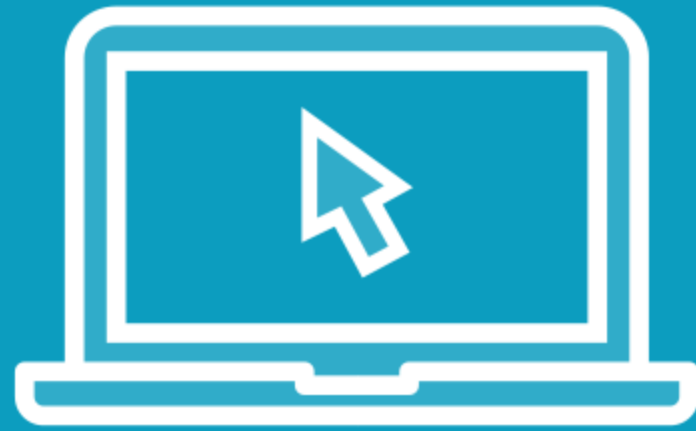
Error Mechanics

Create

`errors.New` and `fmt.Errorf`



Demo



Error Mechanics

comma error pattern

always last parameter



Errors versus Panics



Don't panic.

Go Proverbs - <https://go-proverbs.github.io/>



Errors vs Panic

Errors

result of an operation

easy to discover

implies that things didn't go to plan

used frequently

Panic

alters control flow

relies on docs and reading code

implies that program is unstable

rare



Demo



Errors vs panics

show two styles of error management for simple function

show how to trap panic (divide by zero?) with `recover()` and convert to error



Summary



Errors in Go

Error Mechanics

Errors vs Panics

