LESSON 7

main.go

```
type Shape interface {
  Area() float64
}
```

Go

Interfaces

Learn it all @MyWebUniversity



A FREE MYWEBUNIVERSITY COURSE

GO PROGRAMMING

LESSON 7

Interfaces and Embedding

Implicit interfaces, any, type switches, embedding and the error interface

Interfaces

shapes.go

```
type Shape interface {
    Area() float64
}
type Rect struct{ W, H float64 }
type Circle struct{ R float64 }
func (r Rect) Area() float64   { return r.W * r.H }
func (c Circle) Area() float64 { return math.Pi * c.R * c.R }

shapes := []Shape{Rect{3, 4}, Circle{1}}
for _, s := range shapes { fmt.Println(s.Area()) }
```

- An interface lists method signatures. Any type that has those methods satisfies the …
- This is called structural typing, and it lets you connect packages that never knew about …

01

Small Interfaces



- io.Reader — Read(p []byte)
- io.Writer — Write(p []byte)
- fmt.Stringer — String() string

02

any and Type Switches

describe.go

```go
func describe(v any) string {
    switch x := v.(type) {
    case int:    return fmt.Sprintf("int %d", x)
    case string: return "string " + x
    case Shape:  return fmt.Sprintf("shape, area %.1f", x.Area())
    default:     return "unknown"
    }
}
s, ok := v.(string)          // type assertion
```

- The empty interface, written any, holds a value of any type. To use it, get the real type …
- Use any sparingly. Specific types and small interfaces keep your code checked by the compiler.

03

Embedding

embed.go

```
type Person struct{ Name string }
func (p Person) Hello() string { return "Salaam, " + p.Name }

type Teacher struct {
    Person                // embedded
    Subject string
}
t := Teacher{Person{"Ahmad"}, "Math"}
fmt.Println(t.Hello(), t.Name)   // promoted
```

- Go has no inheritance. Instead, embed one struct inside another: its fields and methods are …
- Interfaces can embed other interfaces too. io dot ReadWriter is simply Reader plus Writer. …

04

The error Interface

errors.go

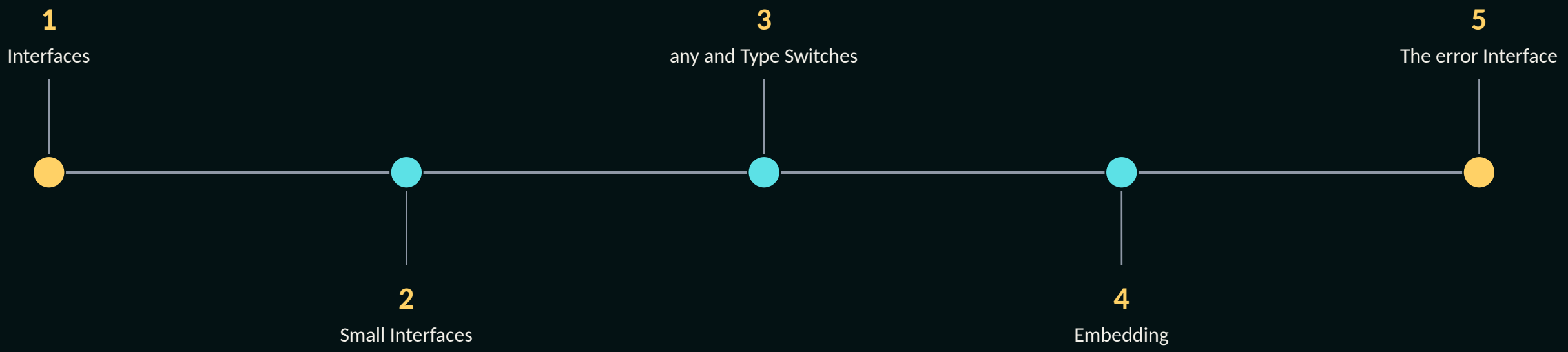
```go
var ErrNotFound = errors.New("not found")

func find(id int) error {
    return fmt.Errorf("find user %d: %w", id, ErrNotFound)
}
err := find(7)
if errors.Is(err, ErrNotFound) {
    fmt.Println(err)     // find user 7: not found
}
```

- error is just an interface with one method: Error, returning a string. Make custom error …
- errors dot Is checks for a specific error in the chain, and errors dot As pulls out a …

05

Lesson Map


1
Interfaces
2
Small Interfaces
3
any and Type Switches
4
Embedding
5
The error Interface

Sources and Further Reading

shapes.go

```go
type Shape interface {
    Area() float64
}
type Rect struct{ W, H float64 }
type Circle struct{ R float64 }
func (r Rect) Area() float64   { return r.W * r.H }
func (c Circle) Area() float64 { return math.Pi * c.R * c.R }

shapes := []Shape{Rect{3, 4}, Circle{1}}
for _, s := range shapes { fmt.Println(s.Area()) }
```


describe.go

```go
func describe(v any) string {
    switch x := v.(type) {
    case int:    return fmt.Sprintf("int %d", x)
    case string: return "string " + x
    case Shape:  return fmt.Sprintf("shape, area %.1f", x.Area())
    default:     return "unknown"
    }
}
s, ok := v.(string)          // type assertion
```

Thank you for watching

MyWebUniversity.com

Free technical education since 2000 — English and Dari

@MyWebUniversity · @MyWebUniversity-Dari