Go Interfaces

WASA: Web and Software Architecture

Enrico Bassetti

Methods

Recap: Struct

```
type Vertex struct {
  X int
  Y int
}

// Somewhere in the code
v := Vertex{1, 2}
fmt.Println(v)
```

Struct methods

```go
type Vertex struct {
  X int
  Y int
}

func (v Vertex) Equal(other Vertex) bool {
  return v.X == other.X && v.Y == other.Y
}

// Somewhere in the code
vtx := Vertex{1, 2}
if vtx.Equal(Vertex{2, 3}) {
// ...
```

Pointer receiver

```go
type Vertex struct {
  X int
  Y int
}

func (v *Vertex) Equal(other *Vertex) bool {
  return v.X == other.X && v.Y == other.Y
}

// Somewhere in the code
vtx := &Vertex{1, 2}
if vtx.Equal(&Vertex{2, 3}) {
// ...
```

Methods on types!

```go
type Latitude float64

func (lat *Latitude) IsValid() bool {
    return lat != nil && -90 <= *lat && *lat <= 90
}

type Longitude float64

func (lng *Longitude) IsValid() bool {
    return lng != nil && -180 <= *lng && *lng <= 180
}
```

Methods on types!

```
type Point2D struct {
    Latitude  Latitude
    Longitude Longitude
}

func (p Point2D) IsValid() bool {
    return p.Latitude.IsValid() && p.Longitude.IsValid()
}

func (p Point2D) IsZero() bool {
    return p.Latitude == 0 && p.Longitude == 0
}
```

Interfaces

A simple interface

```
// fmt/print.go:62
type Stringer interface {
    String() string
}
```

Struct implementing Stringer

```go
type Vertex struct {
  X int
  Y int
}

func (v *Vertex) String() string {
  return fmt.Sprintf("(%d,%d)", v.X, v.Y)
}

// ...

func printSomething(s fmt.Stringer) {
  fmt.Println(s.String())
}
```

Writer

```go
// io/io.go:96
type Writer interface {
    Write(p []byte) (n int, err error)
}
// net/http/server.go:95
type ResponseWriter interface {
    Header() Header
    Write([]byte) (int, error)
    WriteHeader(statusCode int)
}

func saveFile(w io.Writer, content []byte) {
  w.Write(content)
}
```