The Type Switch

In this lesson, you'll learn how to use the switch statement to determine the type of interface and writing cases accordingly.

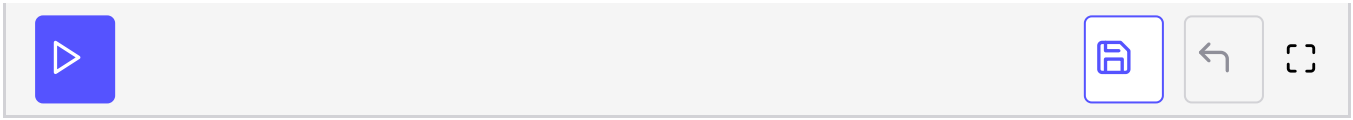
WE'LL COVER THE FOLLOWING

Testing the type of interface

The type of an interface variable can also be tested with a special kind of switch: the *type-switch*. Look at the following program:

```
package main
import (
  "fmt"
  "math"
)

type Square struct {
  side float32
}

type Circle struct {
  radius float32
}

type Shaper interface {
  Area() float32
}

func main() {
  var areaIntf Shaper
  sq1 := new(Square)
  sq1.side = 5
  areaIntf = sq1

  switch t := areaIntf.(type) {
    case *Square:
      fmt.Printf("Type Square %T with value %v\n", t, t)

    case *Circle:
      fmt.Printf("Type Circle %T with value %v\n", t, t)

    default:
```

```
    default:
      fmt.Printf("Unexpected type %T", t)
    }
}

func (sq *Square) Area() float32 {
  return sq.side * sq.side
}

func (ci *Circle) Area() float32 {
  return ci.radius * ci.radius * math.Pi
}
```



The Type Switch

The variable `t` receives both value and type from `areaIntf`. All of the listed types have to implement the interface (`Shaper` in this case); if the current type is none of the case-types, the default clause is executed. *Fallthrough* is not permitted. With a type-switch, a runtime type analysis can be done. Of course, all the built-in types as *int*, *bool*, and *string* can also be tested in a type switch.

Never use `element.(type)` outside of a switch statement.

In the following code snippet, a type classifier function is shown which accepts an array with a variable number of arguments of any type and executes something according to the determined type:

```
func classifier(items ...interface{}) {
  for i, x := range items {
    switch x.(type) {
      case bool: fmt.Printf("param #%d is a bool\n", i)
      case float64: fmt.Printf("param #%d is a float64\n", i)
      case int, int64: fmt.Printf("param #%d is an int\n", i)
      case nil: fmt.Printf("param #%d is nil\n", i)
      case string: fmt.Printf("param #%d is a string\n", i)
      default: fmt.Printf("param #%d's type is unknown\n", i)
    }
  }
}
```

For example, this function could be called as a **classifier(13, -14.3, “BELGIUM”, complex(1, 2), nil, false)**. When dealing with data of an unknown type from external sources, type testing and conversion to Go data types can be very useful, e.g. parsing data that are JSON- or XML-encoded.

That’s it about checking the type of an interface variable with a type-switch; in the next lesson, you have to write a program to solve a problem.