Structs

This lesson explains how to define new types using Structs using an example

WE'LL COVER THE FOLLOWING

Definition

We previously discussed the in built types in Go, we will now move on to look at how new types can be defined using structs. A **struct** is a collection of fields/properties. You can define new types as structs or interfaces. If you are coming from an object-oriented background, you can think of a struct to be a light class that supports composition but not inheritance. Methods are discussed at length in Chapter 5.

You don't need to define getters and setters on struct fields, they can be accessed automatically. However, note that only exported fields (capitalized) can be accessed from outside of a package.

A struct literal sets a newly allocated struct value by listing the values of its fields. You can list just a subset of fields by using the `"Name:" syntax` (the order of named fields is irrelevant when using this syntax). The special prefix `&` constructs a pointer to a newly allocated struct.

Examples

Given below is an example struct declaration:

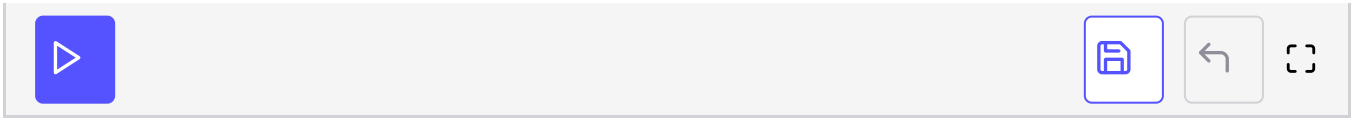
```
package main

import (
        "fmt"
```

```
        "time"
)

type Bootcamp struct {
        // Latitude of the event
        Lat float64
        // Longitude of the event
        Lon float64
        // Date of the event
        Date time.Time
}

func main() {
        fmt.Println(Bootcamp{
                Lat:  34.012836,
                Lon:  -118.495338,
                Date: time.Now(),
        })
}
```



Declaration of struct literals:

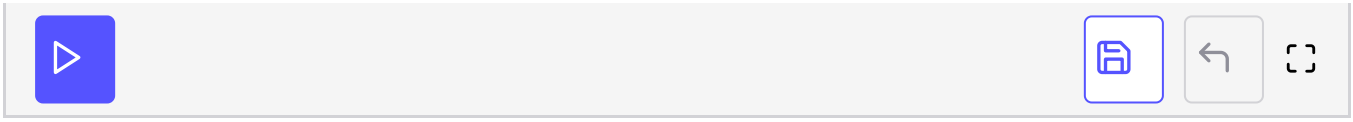
```
package main

import "fmt"

type Point struct {
        X, Y int
}

var (
        p = Point{1, 2}  // has type Point
        q = &Point{1, 2} // has type *Point
        r = Point{X: 1}  // Y:0 is implicit
        s = Point{}      // X:0 and Y:0
)

func main() {
        fmt.Println(p, q, r, s)
}
```



Accessing fields using the dot notation:

```
package main

import (
        "fmt"
        "time"
```

```
)

type Bootcamp struct {
        Lat, Lon float64
        Date     time.Time
}

func main() {
        event := Bootcamp{
                Lat: 34.012836,
                Lon: -118.495338,
        }
        event.Date = time.Now()
        fmt.Printf("Event on %s, location (%f, %f)",
                event.Date, event.Lat, event.Lon)

}
```

Quiz

Quiz on Structs

Q Which of the following is NOT true regarding structs?

COMPLETED 0%

1 of 1

Now that we have learnt of all the different types in Go, we can move on to

discuss how variables can be initialized.