Times and Dates

In this lesson, you'll study the time package and the functions supported by it.

WE'LL COVER THE FOLLOWING

- The `time` Package

The `time` Package

The package **time** gives us a datatype **Time** (to be used as a value) and functionality for displaying and measuring time and dates. The current time is given by `time.Now()`, and the parts of a time can then be obtained as `t.Day()`, `t.Minute()`, and so on. You can make your own time-formats as in:

```
t := time.Now()
fmt.Printf("%02d.%02d.%4d\n", t.Day(), t.Month(), t.Year
()) // e.g.: 29.10.2019
```

The type **Duration** represents the *elapsed time* between two instants as an int64 *nanosecond* count. A handy function is `Since(t Time)` that returns the *time elapsed* since `t`. The type **Location** maps time instants to the *zone* in use at that time. **UTC** represents **U**niversal **C**oordinated **T**ime. There is a predefined function `Format` with syntax as:

```
func (t Time) Format(s string) string
```

It formats a time `t` into a string according to an `s` string, with some predefined formats like `time.ANSIC` or `time.RFC822`. The general layout defines the format by showing the representation of a standard time, which is then used to describe the time to be formatted; this seems strange, but an

example makes this clear:

```
t := time.Now().UTC()
fmt.Println(t.Format("02 Jan 2006 15:04"))// e.g: 29 Oct 2019 11:00
```

Run the following program to see how the above-discussed functionalities work.

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

var week time.Duration

func main() {
    t := time.Now()
    fmt.Println(t)
    fmt.Printf("%02d.%02d.%4d\n", t.Day(), t.Month(), t.Year())
    t = time.Now().UTC()
    fmt.Println(t)
    fmt.Println(time.Now())
    // calculating times:
    week = 60 * 60 * 24 * 7 * 1e9 // must be in nanosec
    week_from_now := t.Add(week)
    fmt.Println(week_from_now)
    // formatting times:
    fmt.Println(t.Format(time.RFC822))
    fmt.Println(t.Format(time.ANSIC))
    fmt.Println(t.Format("02 Jan 2006 15:04"))
    s := t.Format("2006 01 02")
    fmt.Println(t, "=>", s)
}
```

Date and Time

As you can see in the above code, at **line 10** we declared `t` time and initialized with the present time by `time.Now()`. Printing the `t` at **line 11** prints complete time with default format. **At line 12** we specify the format of `t` as **dd.mm.yyyy**. Next, we decide to add *week* in our time. We declared the `week` variable and initialize it with a value of $(60 * 60 * 24 * 7 * 1e9)$ at **line 17**. We multiply a factor of $1e9$ because we have to keep it in nanoseconds. At **line 18** we add `week` to our time `t`. Now after printing `t`, change is noticeable, as weeks are added into `t`. At **line 21** time is printed in `RFC822` format, e.g., `30 Oct 19 11:34 UTC`. At **line 22** time is printed in `ANSIC` format, e.g., `Wed Oct 30`

`11:34:03 2019`. At **line 23** time is printed in `02 Jan 2006 15:04` format, e.g., `30 Oct 2019 11:34`. At **line 24** we create a format(yyyy.mm.dd) and store it in `s`. In the next line, we print the time `t` in default format and then in the format specified above.

For more information on Golang's `time` package, visit this page.

That's about `time` package. In the next lesson, you'll study *pointers*.