TutorialKart

Golang Map – Declare, Initialize, Access, Iterate, Update, Delete

In Golang, Map is a collection of unordered key:value pairs. While key has to be distinct in a Map, value can occur in duplicates.

You can access the key:value pair of a Map using key. **Key** in a Map acts like an **index** in an Array.

Golang Map Declaration

To declare a Golang map, you have to use `map` keyword along with the datatypes of key and value.

```
var map_name map[key_data_type]value_data_type
```

The square brackets are mandatory and does not represent an optional.

example.go

```
package main

import "fmt"

func main() {
    var colorMap map[string]string
    fmt.Println(colorMap)
}
```

Output

```
map[]
```

Declaring a map creates an empty map.

Golang Map Initialization

To initialize a Golang map, you can use assignment operator and a set of all key value pairs.

```
var map_name = map[key_data_type]value_data_type{key1:value1, key2:value2}
```

```
var map_name = map[key_data_type]value_data_type {key1:value1, key2:value2}
```

The square brackets are mandatory and does not represent an optional.

example.go

```
package main

import "fmt"

func main() {
    var colorMap = map[string]string {"white":"#FFFFFF", "black":"#000000", "red":"#FF00
    fmt.Println(colorMap)
}
```

Output

```
map[white:#FFFFFF black:#000000 red:#FF0000 blue:#0000FF green:#00FF00]
```

You can also add a key:value pair to the Map using key as index and value as element.

example.go

```
package main

import "fmt"

func main() {
    var colorMap = make(map[string]string)
    colorMap["white"] = "#FFFFFF"
    colorMap["black"] = "#000000"
    colorMap["red"] = "#FF0000"
    colorMap["blue"] = "#0000FF"
    colorMap["green"] = "#00FF00"

    fmt.Println(colorMap)
}
```

Output

```
map[green:#00FF00 white:#FFFFFF black:#000000 red:#FF0000 blue:#0000FF]
```

Observe that the key value pairs are printed not in an order, because Map is a non-sequential collection.

Access Key:Value pairs of Map

You can get Value using Key from a Map. Key can be used as an index.

example.go

```
package main

import "fmt"

func main() {
    var colorMap = map[string]string {"white":"#FFFFFF", "black":"#000000", "red":"#FF00

    fmt.Println(colorMap["white"])
    fmt.Println(colorMap["red"])
}
```

Output

```
#FFFFFF
#FF0000
```

Iterate through Key:Value pairs of Map

You can iterate through key value pairs of a Map using Range.

example.go

```
package main

import "fmt"

func main() {
    var colorMap = map[string]string {"white":"#FFFFFF", "black":"#000000", "red":"#FF00

    for key, value := range colorMap {
        fmt.Println("Hex value of", key, "is", value)
    }
}
```

Output

```
Hex value of white is #FFFFFF
Hex value of black is #000000
Hex value of red is #FF0000
Hex value of blue is #0000FF
Hex value of green is #00FF00
```

Update Value for a Key in Map

You can update a value for a Key in Map using assignment operator. Just assign the new value to the Map

indexed by the specific Key.

example.go

```
package main

import "fmt"

func main() {
    var colorMap = map[string]string {"white":"#FFFFFF", "black":"#000000", "red":"#FF00

    // update
    colorMap["red"] = "#FF2222"

    fmt.Println(colorMap["red"])
}
```

Output

```
#FF2222
```

Delete a Key:Value pair from a Golang Map

To delete a Key:Value pair, you can use delete() function.

```
delete(map_name, Key)
```

In the following example, we will delete the key:value pairs for the Keys:[“white”, “black”].

example.go

```
package main

import "fmt"

func main() {
    var colorMap = map[string]string {"white":"#FFFFFF", "black":"#000000", "red":"#FF00

    delete(colorMap, "white")
    delete(colorMap, "black")

    fmt.Println(colorMap)
}
```

Output

```
map[blue:#0000FF green:#00FF00 red:#FF0000]
```

Conclusion

In this Golang Tutorial, we learned about Golang Maps and about different operations to read and update the elements of Golang Map.