Golang Session

- Harsh Dusane

Topic : Arrays

Arrays

- An array is a data structure that consists of a collection of elements of a single type or simply you can say a special variable, which can hold more than one value at a time. The values an array holds are called its elements or items. An array holds a specific number of elements, and it cannot grow or shrink. Different data types can be handled as elements in arrays such as Int, String, Boolean, and others. The index of the first element of any dimension of an array is 0, the index of the second element of any array dimension is 1, and so on.

Declaring an Integer or String Array of Five Elements in Go

```
package main
import (
        "fmt"
        "reflect"
)
func main() {
        var intArray [5]int
        var strArray [5]string
        fmt.Println(reflect.ValueOf(intArray).Kind())
        fmt.Println(reflect.ValueOf(strArray).Kind())
}
```

How to assign and access array element values in Go?

```
package main
import "fmt"
func main() {
        var theArray [3]string
        theArray[0] = "India"  // Assign a value to the first element
        theArray[1] = "Canada" // Assign a value to the second element
        theArray[2] = "Japan"  // Assign a value to the third element
        fmt.Println(theArray[0]) // Access the first element value
        fmt.Println(theArray[1]) // Access the second element valu
        fmt.Println(theArray[2]) // Access the third element valu
}
```

How to initialize an Array with an Array Literal in Go?

```
package main
import "fmt"
func main() {
        x := [5]int{10, 20, 30, 40, 50}   // Intialized with values
        var y [5]int = [5]int{10, 20, 30} // Partial assignment
        fmt.Println(x)
        fmt.Println(y)
}
```

Initializing an Array with ellipses in Go

```
package main
import (
        "fmt"
        "reflect"
)
func main() {
        x := [...]int{10, 20, 30}
        fmt.Println(reflect.ValueOf(x).Kind())
        fmt.Println(len(x))
}
```

Initialize values for specific array elements in Go

```
package main
import "fmt"
func main() {
        x := [5]int{1: 10, 3: 30}
        fmt.Println(x)
}
```

How to iterate over an Array using for loop?

```
package main
import "fmt"
func main() {
	intArray := [5]int{10, 20, 30, 40, 50}
	fmt.Println("\n---------------Example 1--------------------\n")
	for i := 0; i < len(intArray); i++ {
		fmt.Println(intArray[i])
	}
	fmt.Println("\n---------------Example 2--------------------\n")
	for index, element := range intArray {
		fmt.Println(index, "=>", element)
	}
	fmt.Println("\n---------------Example 3--------------------\n")
	for _, value := range intArray {
		fmt.Println(value)
	}
	j := 0
	fmt.Println("\n---------------Example 4--------------------\n")
	for range intArray {
		fmt.Println(intArray[j])
		j++
	}
}
```

Copy an array by value and reference into another array

```
package main
import "fmt"
func main() {
        strArray1 := [3]string{"Japan", "Australia", "Germany"}
        strArray2 := strArray1  // data is passed by value
        strArray3 := &strArray1 // data is passed by refrence
        fmt.Printf("strArray1: %v\n", strArray1)
        fmt.Printf("strArray2: %v\n", strArray2)
        strArray1[0] = "Canada"

        fmt.Printf("strArray1: %v\n", strArray1)
        fmt.Printf("strArray2: %v\n", strArray2)
        fmt.Printf("*strArray3: %v\n", *strArray3)
}
```

Golang check if array element exists

```
package main
import (
        "fmt"
        "reflect"
)
func main() {
        strArray := [5]string{"India", "Canada", "Japan", "Germany", "Italy"}
        fmt.Println(itemExists(strArray, "Canada"))
        fmt.Println(itemExists(strArray, "Africa"))
}
func itemExists(arrayType interface{}, item interface{}) bool {
        arr := reflect.ValueOf(arrayType)
        if arr.Kind() != reflect.Array {
                panic("Invalid data-type")
        }
        for i := 0; i < arr.Len(); i++ {
                if arr.Index(i).Interface() == item {
                        return true
                }
        }
        return false
}
```

Tricks to filter array elements in Go

```
package main
import "fmt"
func main() {
        countries := [...]string{"India", "Canada", "Japan", "Germany", "Italy"}
        fmt.Printf("Countries: %v\n", countries)
        fmt.Printf(":2 %v\n", countries[:2])
        fmt.Printf("1:3 %v\n", countries[1:3])
        fmt.Printf("2: %v\n", countries[2:])
        fmt.Printf("2:5 %v\n", countries[2:5])
        fmt.Printf("0:3 %v\n", countries[0:3])
        fmt.Printf("Last element: %v\n", countries[len(countries)-1])
        fmt.Printf("All elements: %v\n", countries[0:len(countries)])
        fmt.Println(countries[:])
        fmt.Println(countries[0:])
        fmt.Println(countries[0:len(countries)])
        fmt.Printf("Last two elements: %v\n", countries[len(countries)-2:len(countries)])
}
```