Swift Enum

Swift Enum

Enum keyword is used for enumerations in Swift programming.

What is enumeration?

By English definition, enumeration mean "**the action of mentioning a number of things one by one**". We do in Swift programming the same thing. We declare a set of named values called elements and each of these elements gets a number implicitly. We use these elements as constants throughout the program.

Syntax of Enum in Swift

The syntax to declare an enum is:

```
enum enumName {
// enumerated
```

```
enum enumName {
  // enumerated elements
}
```

where `enumName` is the name by which we can reference this enum.

Inside the curly braces, you can declare the elements, one for each line. For each element you declare, you should provide the keyword case prior to the element name. An example is shown below.

```
enum enumName {
case Element1
```

```
enum enumName {
  case Element1
  case Element2
}
```

You can declare as many number of elements inside Enum as per requirement.

Example – Swift Enum

Following is a simple example, where we have an enum to declare days of week as elements. We use switch case to demonstrate the usage of enum.

main.swift

```
enum WeekDay {
```

```
enum WeekDay {
   case Monday
   case Tuesday
   case Wednesday
   case Thursday
   case Friday
   case Saturday
   case Sunday
}

func action(day: WeekDay){
    switch day{
        case .Monday:
            print("Monday is working day. Go to office.")
        case .Tuesday:
            print("Tuesday is working day. Go to office.")
        case .Wednesday:
            print("Wednesday is working day. Go to office.")
        case .Thursday:
            print("Thursday is working day. Go to office.")
        case .Friday:
            print("Friday is status day. Prepare stutus presentation for office.")
        case .Saturday:
            print("Saturday is off day. Rest at home.")
        case .Sunday:
            print("Sunday is off day. Rest at home.")
    }
}

let today = WeekDay.Thursday
action(day:today)

action(day:WeekDay.Friday)
action(day:WeekDay.Saturday)
```

Output

```
Thursday is working
```

```
Thursday is working day. Go to office.
Friday is status day. Prepare stutus presentation for office.
Saturday is off day. Rest at home.
```

Conclusion

In this Swift Tutorial, we learned the syntax and usage of Enumerations in Swift programming.

- Swift Dictionary
- Swift - Create Dictionary
- Swift - Create Dictionary using Arrays
- Swift - Iterate through Dictionary
- Swift - Get Dictionary Size
- Swift - Check if Dictionary is Empty
- Swift - Add or Append Element to Dictionary
- Swift - Get value using key in Dictionary
- Swift - Check if a key is present in Dictionary
- Swift - Merge Two Dictionaries
- Swift - Convert Dictionary into Arrays of Keys and Values
- Swift - Print all Keys in a Dictionary

Sets

- Swift Print Set
- Swift Get Set Size
- Swift Insert Element to Set
- Swift Check if Element is present in Set

File Operations

- Swift Read Text File

Swift Errors [Solved]

- Swift error: return from initializer without initializing all stored properties
- Swift – struct error: missing argument labels in call