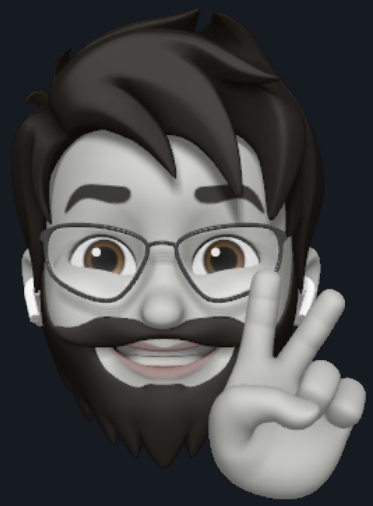
QUICK TIPS : iOS

WHAT IS ASSOCIATED & OPAQUE TYPE IN SWIFT?



Rohit Saini
iOS Developer

QUICK TIPS : iOS

ASSOCIATED TYPE:

- An *associated type* gives a placeholder name to a type that is used as part of the protocol.
- The actual type to use for that associated type isn't specified until the protocol is adopted.
- Associated types are specified with associatedtype keyword.

Example:-

```
protocol User{
    associatedtype UserType
    var id: UserType {get}
    func description() -> String
}
```

UserType in the above example can be anything like Int, String, Enum etc.

QUICK TIPS : iOS

```
struct Instagram: User{
    typealias UserType = String
    var id:String
    func description() -> String{
        return "This is a Instagram User"
    }
}

struct Facebook: User{
    typealias UserType = Int
    var id:Int
    func description() -> String{
        return "This is a Facebook User"
    }
}
```

- We have created two struct Instagram and Facebook and conforms to User protocol that we have created earlier.
- In Instagram struct our UserType is a String and Facebook UserType is an Int.
- AssociatedType allows us to make our protocol more powerful with generic feature.
- AssociatedType allows us to use protocol properties with different data Types.

QUICK TIPS : iOS

OPAQUE TYPE(some keyword):

- We can hide the concrete return type of a computed property or function.
- An opaque types always refers to one specific, concrete type - you just don't know which one.

Example:-

```
struct SocialUser{
    func makeUser() -> User{
        return Facebook(id: 9)
    }
}
```

- We have created a struct SocialUser and inside that we want to build a function that produces a user.
- We don't care what kind of user, so we're using the User protocol as its return type.
- When you run the code, you will get compile-time error that says Protocol 'User' can only be used as a generic constraint because it has Self or associated type requirements.

QUICK TIPS : iOS

SOME KEYWORD:

We can use the some keyword to create an Opaque type and resolve this error.

```
struct SocialUser{
    func makeUser() → some User{
        return Facebook(id: 9)
    }
}

          or

struct SocialUser{
    func makeUser() → some User{
        return Instagram(id: "StringUser")
    }
}
```

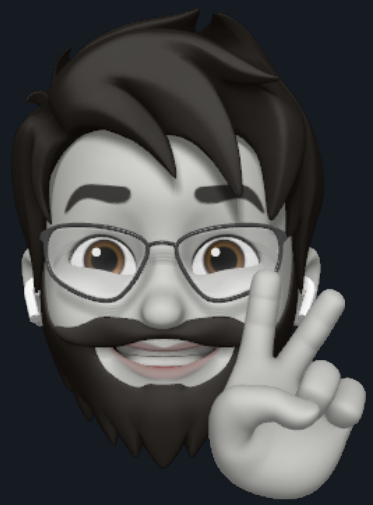
- Now our makeUser function knows that it can now return *a* type that conforms to the User protocol - always the same one, but we don't know which one.
- Now we can return Instagram, Facebook because they both conforms to User protocol.

QUICK TIPS : iOS

LIKE, SHARE, ENJOY

CONNECT.IOS

Scan to Connect



Rohit Saini
iOS Developer