Chapter IX

Notification

VIII.Notification in iOS

1. Notification

Local and push notifications, called remote notification, are great for keeping users informed with timely and relevant content, whether your app is running in the background or inactive. Notifications can display a message, play a distinctive sound, or update a badge on your app icon and our app may take different actions, such as downloading new data.

-Inter-App Notification: **Using NSNotificationCenter to trigger events**

-**Push Notification**: arrived from outside the device and is pushed to your app , originated from your remote server , on a user's device via the Apple Push Notification service ,(APNs).

-**Local Notification**: scheduled and sent by the app itself, without necessary involvement of the server/Internet.

1. Inter-App Notification

A notification dispatch mechanism that enables the broadcast of information to registered observers.

Each running app has a `default` notification center, and you can create new notification centers to organize communications in particular contexts.A notification center can deliver notifications only within a single program; if you want to post a notification to other processes or receive notifications from other processes, use DistributedNotificationCenter instead.

Note : Preparation In App Delegate

```
import UserNotifications
```

And Modify app delegate class as:

```
class AppDelegate: UIResponder, UIApplicationDelegate,
UNUserNotificationCenterDelegate {
```

Step1 : Registering Events

```
NotificationCenter.default.addObserver(self, selector:
#selector(nameOfFunction), name: Notification.Name.init(rawValue:
"NotificationNameToObserve"), object: nil)

//In func NameofFunction

@objc func nameOfFunction (_ noti:Notification) {
    if let object  = noti.object as? TYPEOFOBJECT {
        //Do something with object
      }
    }
  }
```

Step2 : Firing Events in Anywhere in App

```
let notiName = Notification.Name.init(rawValue:
"NotificationNameToObserve")
 NotificationCenter.default.post(name: notiName, object:  ANYOBJECT )
```

2. Push Notification

Step1 : Registering Push Notification

```
let center = UNUserNotificationCenter.current()

 center.delegate = self
 let authOptions:UNAuthorizationOptions = [.alert, .sound, .badge  ]
 center.requestAuthorization(options: authOptions) { (status, error) in
     print(status,error)
     if status {

         UIApplication.shared.registerForRemoteNotifications()
     }
 }
```

Note: Fall back

```
func application(_ application: UIApplication,
    didFailToRegisterForRemoteNotificationsWithError error: Error) {
        print("Unable to register for remote notifications: \(error.localizedDescription)")
}
```

Step2 : Confirming Push Noti & Getting Token

```
func application(_ application: UIApplication,
```

didRegisterFor

```
RemoteNotificationsWithDeviceToken deviceToken: Data) {

    print(deviceToken)
    let tokenString = deviceToken.reduce("", {
                        $0 + String(format: "%02X", $1)   })
                print(tokenString)

                //**** Send token to server
                Messaging.messaging( ).apnsToken =  deviceToken //Firebase
        }
```

Step3 : Getting Notification Events in App

(Responding when user click alert , while app's state is in

background or inactive or foreground)

```
func application(_ application: UIApplication,

    didReceiveRemoteNotification userInfo: [AnyHashable : Any],
    fetchCompletionHandler completionHandler: @escaping
    (UIBackgroundFetchResult) -> Void) {

        print("Got noti")

        //Parse the notification payload
        //Firebase or server to be inform that received
        //Messaging.messaging( ).appDidReceiveMessage(userInfo)


        //Checking the app state
      print(application.applicationState == .active ? "Active" : "")
      print(application.applicationState == .inactive ? "Inactive" : "")
      print(application.applicationState == .background ? "Background" : "")
      }
```

func userNotificationCenter(_ center: UNUserNotificationCenter, didReceive response: UNNotificationResponse, withCompletionHandler completionHandler: @escaping () -> Void) {

completionHandler( )
}

Step4 : Parsing Payload of Notification in App

```
let aps = userInfo["aps"] as? [String:Any]

let value = aps["KEY"] as? TYPEOFVALUE
```

Step5 : Getting Notification while app is in FOREGROUND

```
func userNotificationCenter(_ center: UNUserNotificationCenter,
   willPresent notification: UNNotification, withCompletionHandler
   completionHandler: @escaping
(UNNotificationPresentationOptions) -> Void) {

    print("Will present got notification”)

    print(notification.date,notification.request.content.title,
notification.request.content.body,notification.request.trigger?.value(for
Key: "contentAvailable"))
       completionHandler([.alert,.sound]) //Carry on
  }
```

Note: Sending push notification required to setup server side or try with third party app together with app setting up itself.

https://itunes.apple.com/sg/app/easy-apns-provider-push-notification-service-testing-tool/id989622350?mt=12

Step 6: Accessories Function

```
//Checking whether running from not

 if let notification =  launchOptions?[.remoteNotification] as?

[String:Any] {  //DO Something }
```

//Resetting icon badge

```
UIApplication.sharedApplication().applicationIconBadgeNumber =
UIApplication.sharedApplication().applicationIconBadgeNumber - 1 }
```

3. Local Notification

1. Local Notification

Step1 : SAME AS PUSH NOTIFICATION //In AppDelegate

Step2 : Scheduling Local Notification

//By Time Interval

//Somewhere in App

```
let content = UNMutableNotificationContent()
content.body = “You are having message”
content.sound = UNNotificationSound.default()
content.categoryIdentifier = "message"

let trigger = UNTimeIntervalNotificationTrigger(timeInterval: 1.0, repeats: false)
let request = UNNotificationRequest(identifier: "question1, content: content, trigger:
trigger)
UNUserNotificationCenter.current().add(request, withCompletionHandler: nil)
```

Step3 : SAME AS PUSH NOTIFICATION

Note:
Actionable notifications let you add custom action buttons to the standard iOS interfaces for local and push notifications. Actionable notifications give the user a quick and easy way to perform relevant tasks in response to a notification. Prior to iOS 8, user notifications had only one default action. In iOS 8 and later, the lock screen, notification banners, and notification entries in Notification Center can display one or two custom actions. Modal alerts can display up to four. When the user selects a custom action, iOS notifies your app so that you can perform the task associated with that action.

//Making Choice in Notification

```
//actions defination
   let action1 = UNNotificationAction(identifier: "ok", title: "OK",
options: [.foreground])
     let category = UNNotificationCategory(identifier: “question1”,
actions: [action1 ], intentIdentifiers: [], options: [])
```

```
UNUserNotificationCenter.current().setNotificationCategories([category])
```

Note: //notification.request.content.title can be check when user press action