Notifications
Notifications are used to provide information to the user based on time or location.

There are several kinds:

- **Basic** *(local)* notifications are sent internally (within the app)
- **Remote** notifications are sent from another device
- **Scheduled local** notifications are sent at a designated time
  - The app determines the schedule (i.e., when the notification should be sent
  - The system handles the actual delivery.
- **Active** notifications interact with the user
Notifications

Notifications are an implementation of the *observer software design pattern*. It follows the same general idea as event-driven and MVC patterns.

- An object maintains a list of *observers* and notifies them when the event they’re registered to receive occurs.
- Notifications are used to implement distributed event-handling.
Key Value Observing (KVO) is a mechanism that allows objects to be notified of changes to specified properties of other objects.

- KVO works with any Swift class, as long as the class inherits from the NSObject class.
- You add an observer for any property you want monitored.
- You implement the `.observeValue(forKeyPath` method.
- You can later remove the observer in `deinit`. 
Basic (Local) Notifications

- A way to notify other parts of your application that something of interest has happened

- 3 steps:
  - Make a call to add an observer and associate a method for a given event (a notification name).
  - Implement a notification handler.
  - Issue a post notification when you want to notify registered observers of a given event.

- Notification handling is synchronous.
  - All observers are executed, in turn, before the post call returns

- You can pass data to the notification handlers
Setting up a property to monitor

KVO is implemented in Objective-C. Any properties you wish to observe need to have the *dynamic* modifier in front to ensure access is dynamically dispatched by Objective-C runtime.

```swift
class MyObservedClass: NSObject {
    dynamic var myObservedProperty = "Hi"

    func updateProperty(newValue: String) {
        myObservedProperty = newValue
    }
}
```
Setting up an observer

Add an observer for any property you want to monitor.

```swift
private var myContext = 0

objectToObserve.addObserver(
    self,
    forKeyPath: "myObservedPropertyChanged",
    options: [.Old, .New],
    context: &myContext)
```

- `objectToObserve` is the object that contains the property we want to monitor.
- The first argument is the object that is registering as the observer.
- `options`: you have the option of receiving the old value, the new value or both.
- `Context`: a unique identifier. (Typically the address of a variable is used)
observeValueForKeyPath

Implement the observeValueForKeyPath method, the method that receives the KVO notification:

```swift
override func observeValue(forKeyPath keyPath: String?,
                         ofObject object: AnyObject?,
                         change: [String : AnyObject]?,
                         context: UnsafeMutablePointer<Void>) {

    // Handle observations only for your context. Pass any others
    // up the chain.
    if context == &myContext {
        // do your stuff here when the value changes
    } else {
        // Pass this up the chain
        super.observeValue(forKeyPath, ofObject: object,
                           change: change,
                           context: context)
    }
}
```
Removing an observer

Make sure to remove the observer using `deinit`.

```swift
deinit {
    // you need to remove the observer before
    // the object goes away
    objectToObserve.removeObserver(self,
        forKeyPath: "myObservedProperty",
        context: &myContext)
}
```
Remote Notifications

- Remote notifications are generated by something outside your application
- They are sent through APNS (Apple Push Notification Server)
- Much more complicated than local notifications
- When remote notifications are received by your device iOS displays them in a little pop-down
- If you touch a remote notification iOS will launch your app (if it isn’t already running) or bring it to the foreground (if it’s in the background) and hand off the notification to you
- You need an Apple Developer ID
Scheduled Local Notifications

To create a scheduled local notification, you use the `UILocalNotification` class combined with an `NSDate` object configured with the date and time that the notification is to be triggered.

- Properties may also be set to specify the text to be displayed to the user, an optional repeat interval, messages and sounds, etc.