Calendar and EventKit
Event Kit

*Event Kit* is a set of classes for accessing and manipulating a user’s calendar events and reminders, which live in the *Event Store* database on a device.

You can, among other things:

- Create a calendar
- Delete a calendar
- Get a list of calendars
- Get the attributes of a given calendar
- Create an event
- Modify an event
- Delete an event
Event Kit

At the heart of EventKit is the class `EKEventStore`.

An instance of `EKEventStore` provides access to an API for performing read and write operations on the user’s calendars and reminder lists.

```swift
let eventStore = EKEventStore()
```
Your app **must** ask for permission to access the calendars and/or reminders.

- You check to see if your app is authorized.
- If it isn’t, you must request access.

It’s important to note that the user can change the calendar access state at any time. Consequently, you’ll want to include this code in `viewWillAppear` to make sure that the **current** state of authorization is used each time the user sees the application interface.
To use Event Kit:

- `import EventKit`
- **Create an instance of** `EKEventStore`
- **Through the** `EKEventStore` **object:**
  - Verify that your app has permission to access the event store
  - Include handling if you don’t have access
- **Read and write calendars / events from and to the event store**
To check to see if your app is authorized to access the event store:

```swift
if (EKEventStore.authorizationStatus(for: .event) != EKAuthorizationStatus.authorized) {
    < handle error >
} else {
    < do stuff >
}
```
If the status returned is *Authorized*, you can start reading and writing from or to the Event Store.

If the status returned is *NotDetermined* (as in the first execution), then ask the user for access to the calendars:

```swift
eventStore.requestAccess(to: .event,
    completion: {(accessGranted: Bool, error: NSError?) in

    if accessGranted == true {
        <we can access the event store>
    } else {
        <help the user give you access>
    }

)}
```
Once you’ve been given access to the calendars, you can get a list of them:

eventStore.calendarsForEntityType(EKEntityType.Event)

This returns an array of EKCalendar objects.
Managing Calendars

Creating calendars:
• **Create an** EKCalendar object
• Set various attributes. (*Source* must be selected from a valid list.)
• After saving, store the key associated with that calendar

Deleting a calendar:
• Get the calendar to delete using the stored key
• Remove the calendar

Creating events:
• Get the calendar you want to add an event to
• **Create an** EKEvent object
• Set various attributes
• Save