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.

```
let eventStore = EKEventStore()
```

Event Kit Authorization

Your app must ask for permission to access the calendars and/or reminders.

- Check to see if your app is authorized:

```
authorizationStatus(
    for entityType: EKEntityType) ->
    EKAuthorizationStatus
```

`entityType`: either `.event` or `.reminder`

returns `EKAuthorizationStatus`:

```
.authorized
.denied
.notDetermined
.restricted
```

Event Kit

- If your app isn't authorized, you must request access.

```
requestAccess(
    to entityType: EKEntityType,
    completion: <completion handler>)
```

`entityType`: either `.event` or `.reminder`
`completion`: code to execute when the request completes.

- Your app is not blocked while the user decides.
- The completion handler executes regardless of what the user's choice was.

Note that the user can change the calendar access state at any time. Consequently, include this code in `viewWillAppear` to make sure that the current state of authorization is used each time the user sees the application interface.

Event Kit

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

Event Kit

To check to see if your app is authorized to access the event store:

```
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:

```
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>
   }

})
```

Event Kit

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.
- 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.

Events

To create an event:

- create an instance of `EKEvent` for the appropriate `eventStore`:

```
let event = EKEvent(eventStore:eventStore)
```

- set the properties of the event:

```
event.title = “UT vs. Oklahoma”
event.startDate = Date(“2019-10-12”)
event.calendar = calendarKey
```