iOS Dev Accelerator
Week 5 day 3

NSFetchResultsController
App States and Multitasking
Region Monitoring

Fetched Results Controller

- “You use a fetched results controller to efficiently manage the results returned from a Core Data fetch request to provide data for a UITableView object”
- A fetched Results controller has 3 modes:
  - No tracking: delegate is set to nil. the controller just shows the data from the original fetch.
  - Memory-only tacking: delegate is non-nil and the file cache name is set to nil. Controller monitors objects in its result set and updates in response to changes.
  - Full persistent tracking: delegate and file cache name are non-nil. Controller tracks and displays changes and maintains a persistent cache of the results.

NSFetchedResultsController

- Add an NSFetchedResultsController property to your table ViewController.
- init() the fetched results controller with 4 parameters:
    - NSFetchRequest - must contain at least one sort descriptor for order.
    - NSManagedObjectContext.
    - optional: a key path that returns the section names.
        - the controller spits the results based on this designated attribute.
    - optional: the name of the cache file the controller should use.

`NSFetchedResultsControllerDelegate`

- the fetched results controller notifies its delegate that the controller's fetched results have been changed due to an add, remove, move, or update operations.
    - `controllerWillChangeContent:` - tell your tableview to beginUpdates.
    - `controllerDidChangeSection:atIndex:forChangeType:`
        - insert new sections or delete old sections in your tableview
    - `controllerDidChangeObject:atIndexPath:ForChangeType:`
        - insert/change/delete/move rows in your tableview
    - `controllerDidChangeContent:` - tell your tableview to endUpdates.

UITableView - Swipe Actions

- `editActionsForRowAtIndexPath:` (tableView delegate)
- Delegate returns an array of actions
- Actions appear in the reverse order as the array you return (FILO)
- Each action has a color, title and actionBlock (closure)

```
let deleteAction = UITableViewRowAction(style: .Default, title: "Delete") {
  (action, indexPath) -> Void in
  //  Do stuff here
}

deleteAction.backgroundColor = UIColor.redColor()
```

NSFetchedResultsController – Demo

App state behaviors

- Apps behave different in the background and foreground.
- The OS purposely limits what your app can do while its in the background, because it always gives resource priority to the foreground app.
- Your app is always notified when it transitions from background to foreground and vice versa.

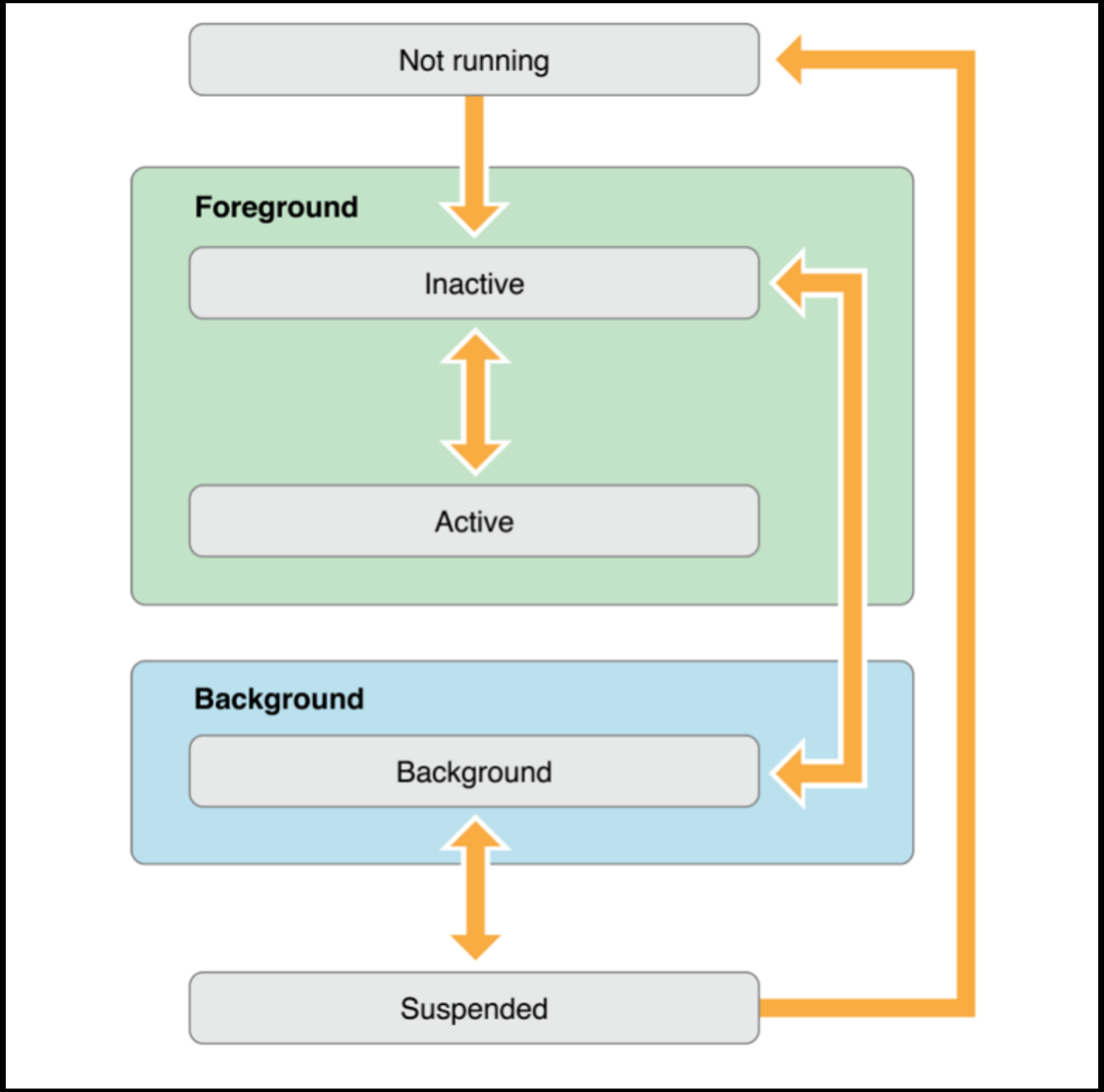
Apples App State Guidelines!

1. Required - Respond appropriately to state transitions in your app.
2. Required - When moving to background specifically, make sure your app adjusts its behavior accordingly.
3. Recommended - Register for notifications that report system changes that your app needs. If your app is suspended, the system queues up these changes and then gives them to you when your app resumes.
4. Optional - If your app actually does work in the background, you need to ask the system for the appropriate permissions to do that work.

App states

- Any at given time, your app is in one of 5 states:
  - **Not Running:** the app has not been launched, or was running and was terminated by the system.
  - **Inactive:** The app is running in the foreground but it is currently not receiving events. This is usually brief and during transitions.
  - **Active:** The app is running in the foreground and receiving events. This is the normal mode for apps.
  - **Background:** The app is in the background and executing code. Most apps briefly enter this mode on their way to being suspended. However, an app that requests extra executing time may remain in this state for an extra period of time. Also, an app being launched directly into the background enters this state instead of inactive.
  - **Suspended:** The app is in the background but not executing code. The system moves apps to this state automatically and does not notify them before doing so. While suspended, an app remains in memory but does not execute code. When a low-memory condition occurs, apps taking up large amounts of memory are terminated.

App states diagram



App state transitions

- Most state transitions are accompanied by a corresponding call to the methods of your app delegate object:
- application:willFinishLaunchingWithOptions: - This method is your apps first chance to execute code at launch time.
- application:DidFinishLaunchingWithOptions: - This method allows you to perform any final initialization before your app is displayed to the user.
- applicationDidBecomeActive: - Lets your app know that it is about to become the foreground app.

App state transitions cont.

- applicationWillResignActive: - Lets you know your app is transitioning away from being the foreground app.
- applicationDidEnterBackground: - Lets you know your app is now running in the background and may be suspended at any time.
- applicationWillEnterForeground: - Lets you know your app is moving out of the background and back into the foreground, but it is not yet active.
- applicationWillTerminate: - Lets you know your app is being terminated. This method is not called if your app is already suspended.

Things to do at Launch time

- Check contents of the launch options dictionary about why the app was launched, and respond appropriately. If its a regular launch from the user you usually don't care, but if the app was launched by the system for some background task, you definitely care.
- Initialize the app’s critical data structures.
- Prepare your app for display if its going to be displayed.
- Remember to keep the willFinishLaunching and didFinishLaunching methods minimal to not slow down launch speeds.
- The system kills any app that takes longer than 5 seconds to launch.
- When launching straight into the background, there shouldn't be much to do except respond to the event that caused the launch.

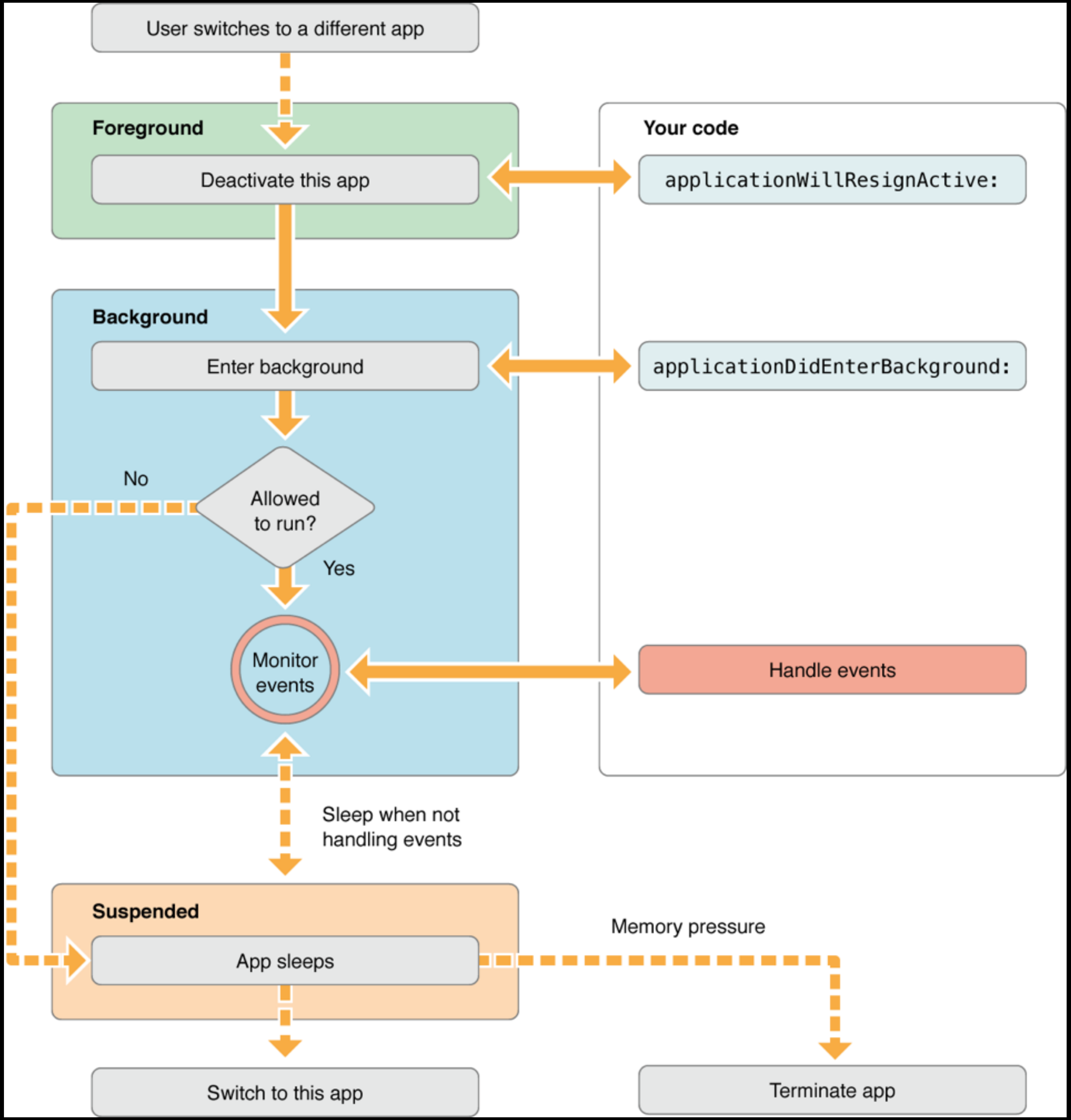
Responding to interruptions

- When an alert based interruption occurs (phone call,text message, system message) your app temporarily moves into the inactive state so the system can send a prompt to a user.
- The app remains in this state until the user dismisses the prompt.
- After the prompt dismisses, your app is either returned to active or move to background state.
- In response to this workflow, your app should do the following things in applicationWillResignActive:
    - Stop timers or periodic tasks
    - Stop running queries
    - Don’t initiate any new tasks.
    - Pause move playback (except when over airplay)
    - Pause the game
    - suspend any non critical operation or dispatch queues.

Moving to the Background

- When user presses home button, sleep/wake button, or launches another app, the foreground app is transitioned to the inactive state and then background state.
- first applicationWillResignActive is called, and then applicationDidEnterBackground.
- Most apps move to suspended state after applicationDidEnterBackground returns or shortly there after.
- Apps that request specific background tasks may continue to run for a while longer.
- New in iOS8 : Location apps are automatically relaunched after the user kills it!

Moving to the Background

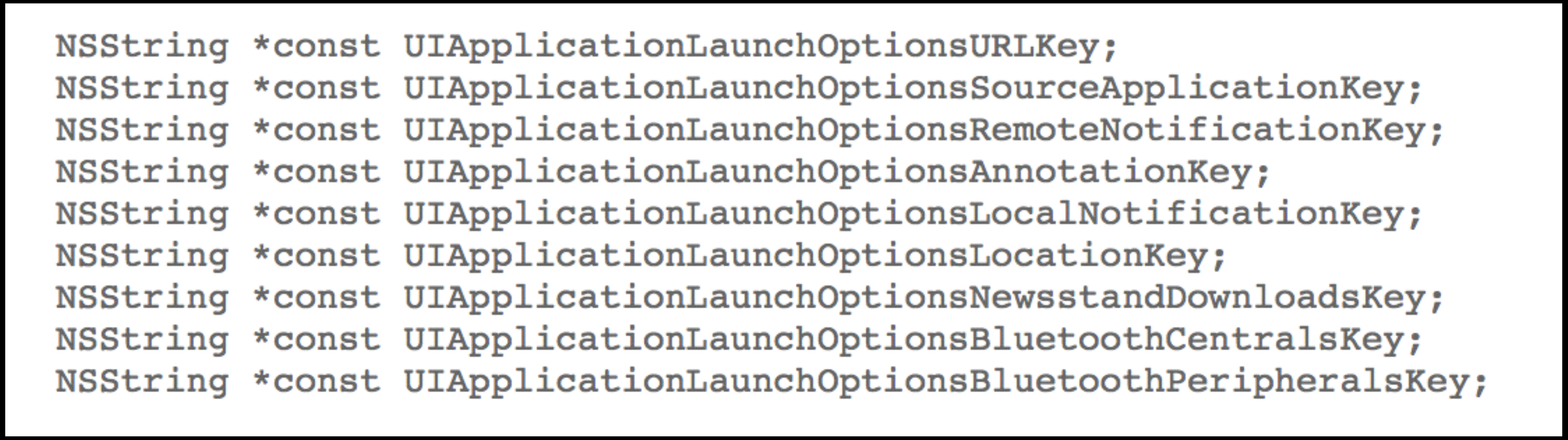


What to do when moving to background

- Say cheese: When the applicationDidEnterBackground method returns, the system takes a picture of your app’s user interface and uses the image for transitions. So clear out any sensitive data.
- Save User Data: all your unsaved data should be saved to disk, since your app could be quietly killed while in the suspended state.
- Free up memory: apps are killed based on how much memory they are taking up, so release as much memory as possible! (big images especially)

DidFinishLaunching Dictionary

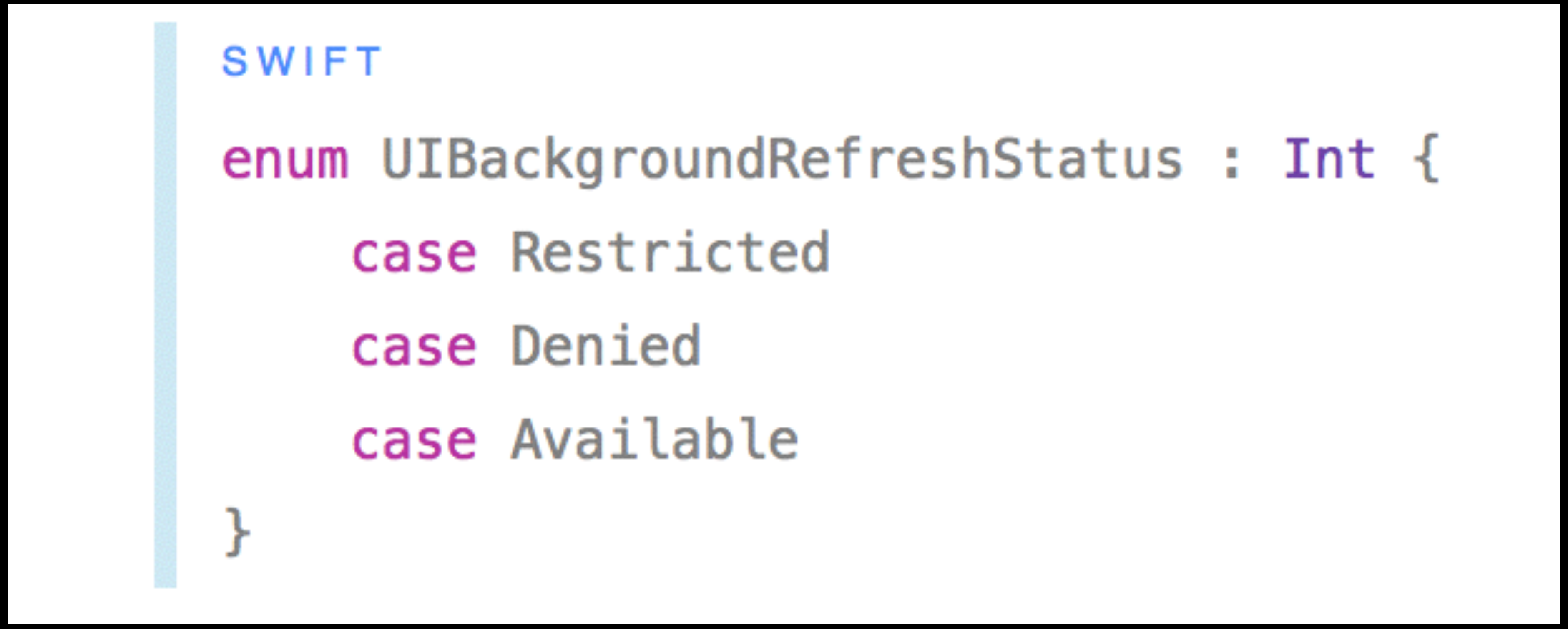
- This dictionary will usually be empty if the user launches the app on their own.



```
NSString *const UIApplicationLaunchOptionsURLKey;
NSString *const UIApplicationLaunchOptionsSourceApplicationKey;
NSString *const UIApplicationLaunchOptionsRemoteNotificationKey;
NSString *const UIApplicationLaunchOptionsAnnotationKey;
NSString *const UIApplicationLaunchOptionsLocalNotificationKey;
NSString *const UIApplicationLaunchOptionsLocationKey;
NSString *const UIApplicationLaunchOptionsNewsstandDownloadsKey;
NSString *const UIApplicationLaunchOptionsBluetoothCentralsKey;
NSString *const UIApplicationLaunchOptionsBluetoothPeripheralsKey;
```

UIApplication.sharedApplication() .backgroundRefreshStatus

- Every app either has the ability to be launched into the background so it can perform background tasks or it doesn't.
- Check this this property to see if this is available, and warn the user if it isn't but your app replies on this.
- Users can disable background refresh on specific apps in Settings>General>Background Refresh


SWIFT

```swift
enum UIBackgroundRefreshStatus : Int {
    case Restricted
    case Denied
    case Available
}
```

Region Monitoring

- “A geographical region is an area defined by a circle of a specified radius around a known point on the Earth’s surface.”
- Apps can use region monitoring to be notified when a user crosses specific boundary.
- “In iOS, regions associated with your app are tracked at all times, included when the app isn't running”
- If it detects a region crossing, your app is relaunched in the background.

Region Monitoring Availability

- Reasons why region monitoring may not be available to the user:
  - hardware not available
  - user denied app authorization for region monitoring
  - user disabled location services in the settings app
  - the user disabled background app refresh in the settings app
  - the device is in airplane mode
- First ask the CLLocationManager class if region monitoring is available with the method isMonitoringAvailableForCass:
- If that returns true, then ask CLLocationManager if location services are enabled with authorizationStatus()

Setting up a Monitor

- In iOS7+ you define geographical regions using the CLCircularRegion class.
- Each region created should include both the data that defines the desired geographical area and a unique identifier string.
- To register a region, call startMonitoringForRegion: method on your CLLocationManager object.
- By default, every time a user's current location crosses a boundary region, the system generates an appropriate region event for your app.
- 2 methods: locationManager:didEnterRegion: and locationManager:didExitRegion: