基本類別與機制

王昱景　Brian Wang
brian.wang.frontline@gmail.com

基本類別

- UIDevice - 代表設備本身
- UIApplication - 代表應用程式本身
- UIScreen - 代表手機螢幕
- UIWindows - 代表主視窗

- UIDevice
  - UIDevice *device = [UIDevice currentDevice];
  - 對多工支援的能力
    - iOS 4.0 / iPhone 3GS 才支援
    - multitaskingSupported：是否支援多功

- 系統資訊
  - name：設備名稱
  - systemName：作業系統名稱
  - systemVersion：作業系統版本
  - model：設備的型號
  - localizedModel：設備的型號 (以本土化訊息顯示)

- userInterfaceIdiom：目前設備的介面形式，iPad (UIUserInterfaceIdiomPad)，iPhone (UIUserInterfaceIdiomPhone)
- uniqueIdentifier：設備 UUID，在 iOS 5 後不被使用
- orientation：設備的方向
  - UIDeviceOrientationUnknown：未知方向
  - UIDeviceOrientationPortrait：肖像模式 (一般手持 iPhone 模式)

- UIDeviceOrientationPortraitUpsideDown：上下顛倒的肖像模式
- UIDeviceOrientationLandscapeLeft：手機往左旋轉的風景模式
- UIDeviceOrientationLandscapeRight：手機往右旋轉的風景模式
- UIDeviceOrientationFaceUp：手機平放且面朝上
- UIDeviceOrientationFaceDown：手機平放且面朝下

- 電池資訊
  - batteryLevel：電池充電狀況，0.0 表示須完全充電，1.0 表示已經 100% 充滿電，存取這個屬性必須確定 batteryMonitoringEnabled 為 YES，否則會回傳 -1
  - batteryMonitoringEnabled：開啟或關閉電池的監控
  - batteryState：顯示電池的狀態
    - UIDeviceBatteryStateUnknown：未知狀態

- UIDeviceBatteryStateUnplugged：電池未充電，連結線已移除
- UIDeviceBatteryStateCharging：電池充電中，電量少於 100%
- UIDeviceBatteryStateFull：電池已 100% 充滿電了

- 距離感測器 (Proximity)
  - proximityMonitoringEnabled：BOOL 值，開啟或關閉距離感測器監測
  - proximityState：BOOL 值，表示距離感測器是否接近使用者，當這個數值變化時系統會發出 UIDeviceProximityStateDidChangeNotification 的訊息

- UIApplication
  - main.m 主程式裡有 UIApplicationMain 這個函數，會回傳 UIApplication 物件，這個物件將一些重要的任務再交付給它的代理者 AppDelegate 取處理
  - 包括產生一個用來作為最底層畫面元件的 UIWindow，並且載入畫面資源檔，再將載入後的 ViewController 附加到 UIWindow 上

- UIAppliction *app = [UIApplication sharedApplication];
- 啟動搖晃時 Undo 功能
  - iOS 3.0 以後才有的功能
  - 輸入後移晃手機，畫面上會出現一個 “還原輸入” 的對話
  - [UIApplication sharedApplication].applicationSupportsShakeToEdit = YES;

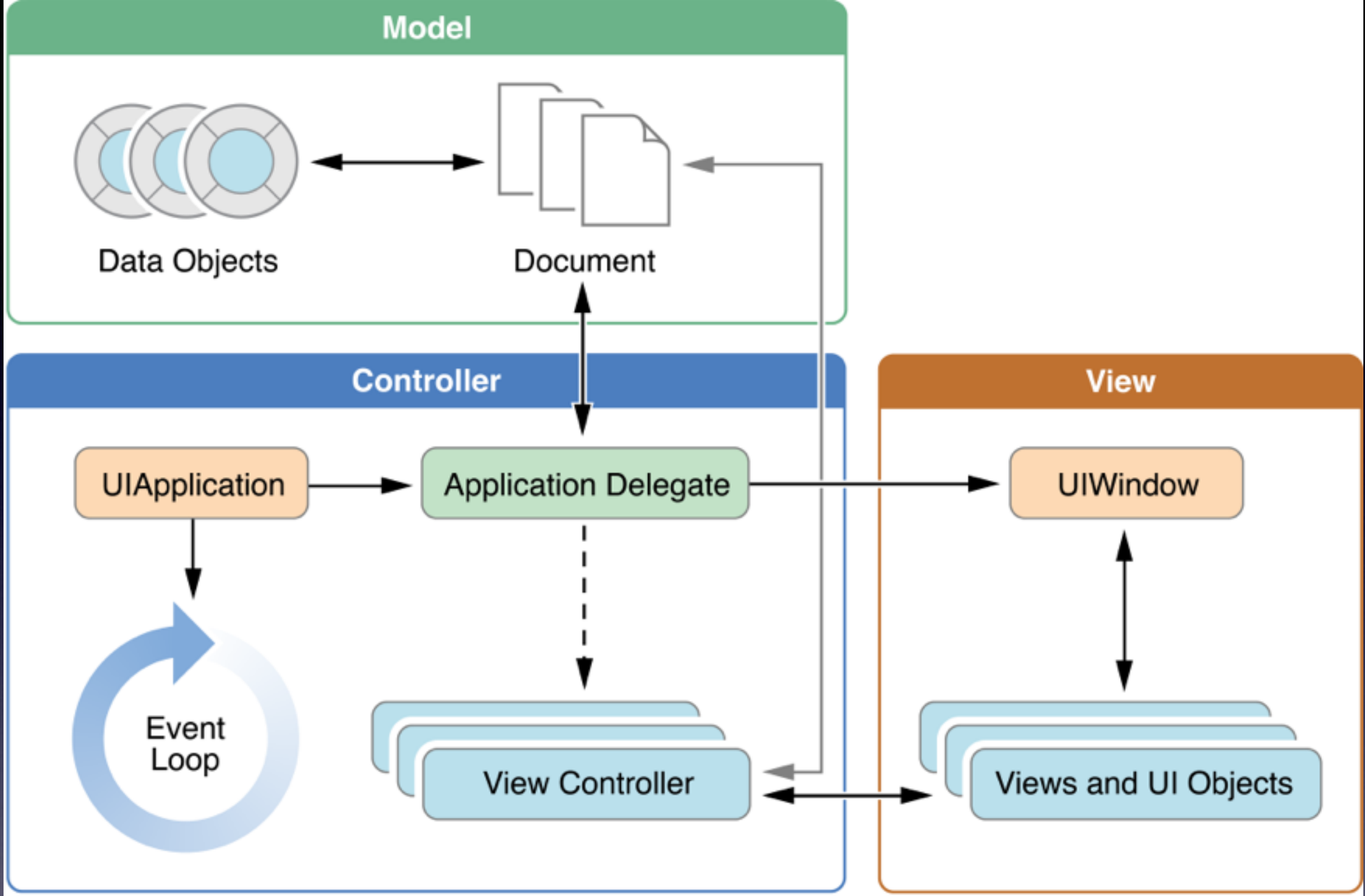
- 取消自動睡眠功能
  - 開啟的應用程式在一段時間後沒動作，手機就會自動進入睡眠狀態以便省電
  - [UIApplication sharedApplication].idleTimerDisable = YES;
- 狀態列的控制
  - iOS 的狀態列是指在畫面最上面的一小塊區域
  - 主要用來顯示訊號、電量等符號

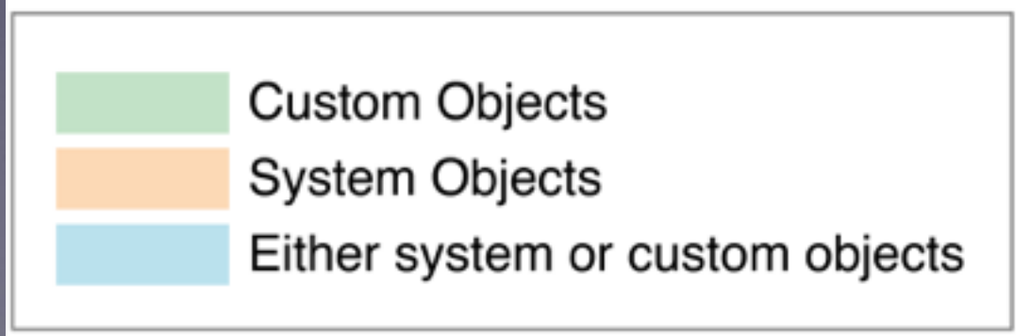
- UIAppliction *app = [UIApplication sharedApplication];
  app.statusBarOrientation = UIInterfaceOrientationLandscapeRight;
  app.statusBarHidden = YES;

- 取得應用程式狀態

```
switch ([UIApplication sharedApplication].applicationState) {
    case UIApplicationStateActive:
        // 喚醒狀態
        break;
    case UIApplicationStateInactive:
        // 睡眠狀態
        break;
    case UIApplicationStateBackground:
        // 背景狀態
        break;
    default:
        break;
}
```

- 註冊遠端與本地端通知
- 設定書報攤 (Newsstand) 內顯示 Icon
- iOS 5 以後才有的應用程式
- - (void)setNewsstandIconImage:(UIImage *)image


Model
Data Objects
Document
Controller
UIApplication
Application Delegate
Event Loop
View Controller
View
UIWindow
Views and UI Objects


Custom Objects
System Objects
Either system or custom objects

- UIScreen
  - 取得螢幕物件的方法
    - NSArray *screen = [UIScreen screens];
    - UIScreen *screen = [UIScreen mainScreen];

- 取得螢幕物件的資訊
  - bounds - CGRect 結構型態，用來表示整個螢幕大小
  - applicationFrame - CGRect 結構型態，用來表示應用程式可顯示畫面的區域
  - scale - 唯讀屬性，用來將邏輯座標轉換為設備實際座標的一個倍率

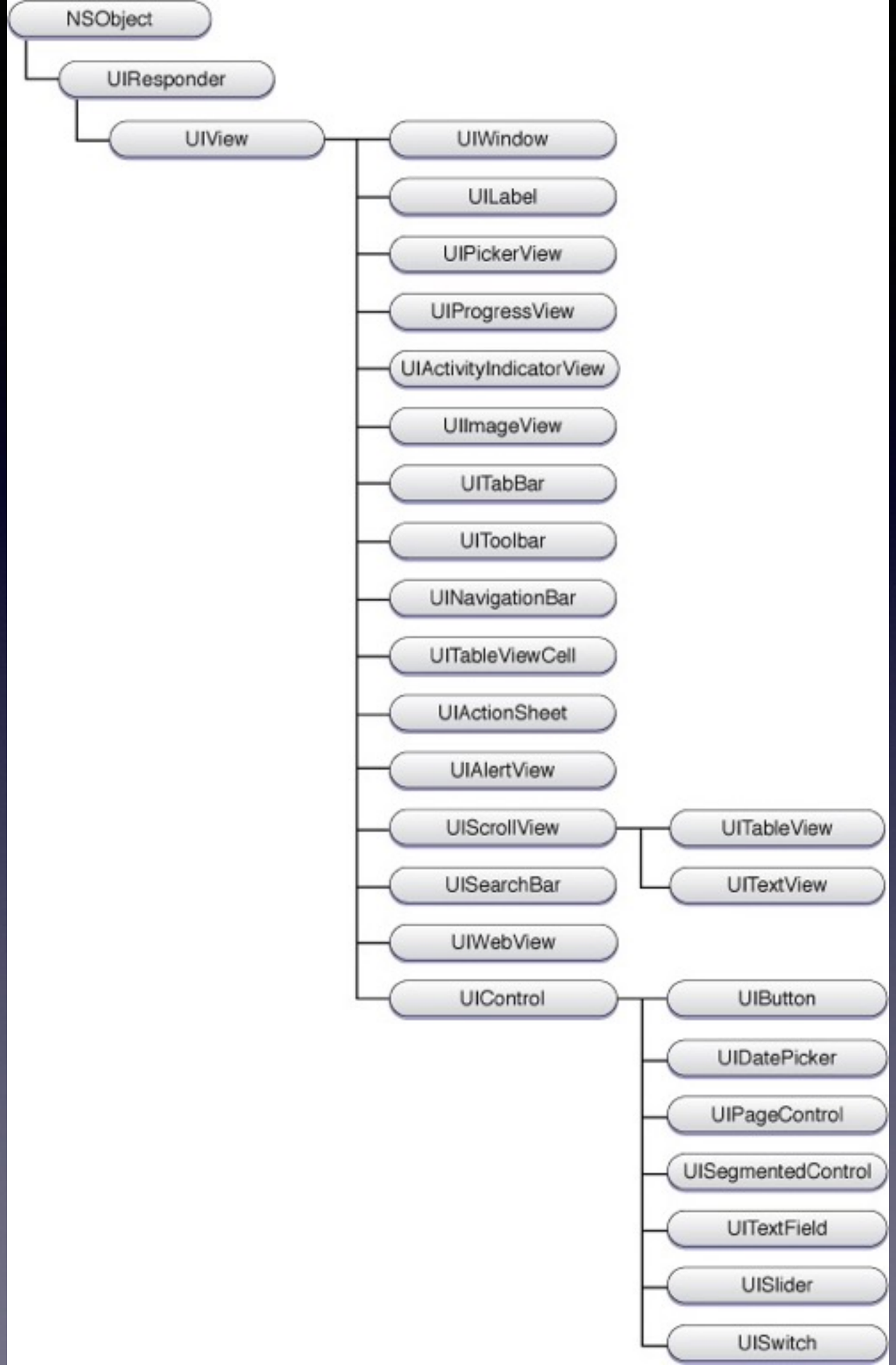
- preferredMode - 唯讀屬性，用來取得偏好顯示模式
- availableModes - 唯讀屬性，用在設備所支援的顯示模式
- currentMode - 目前使用的顯示模式
- 設定螢幕明暗度
  - iOS 5

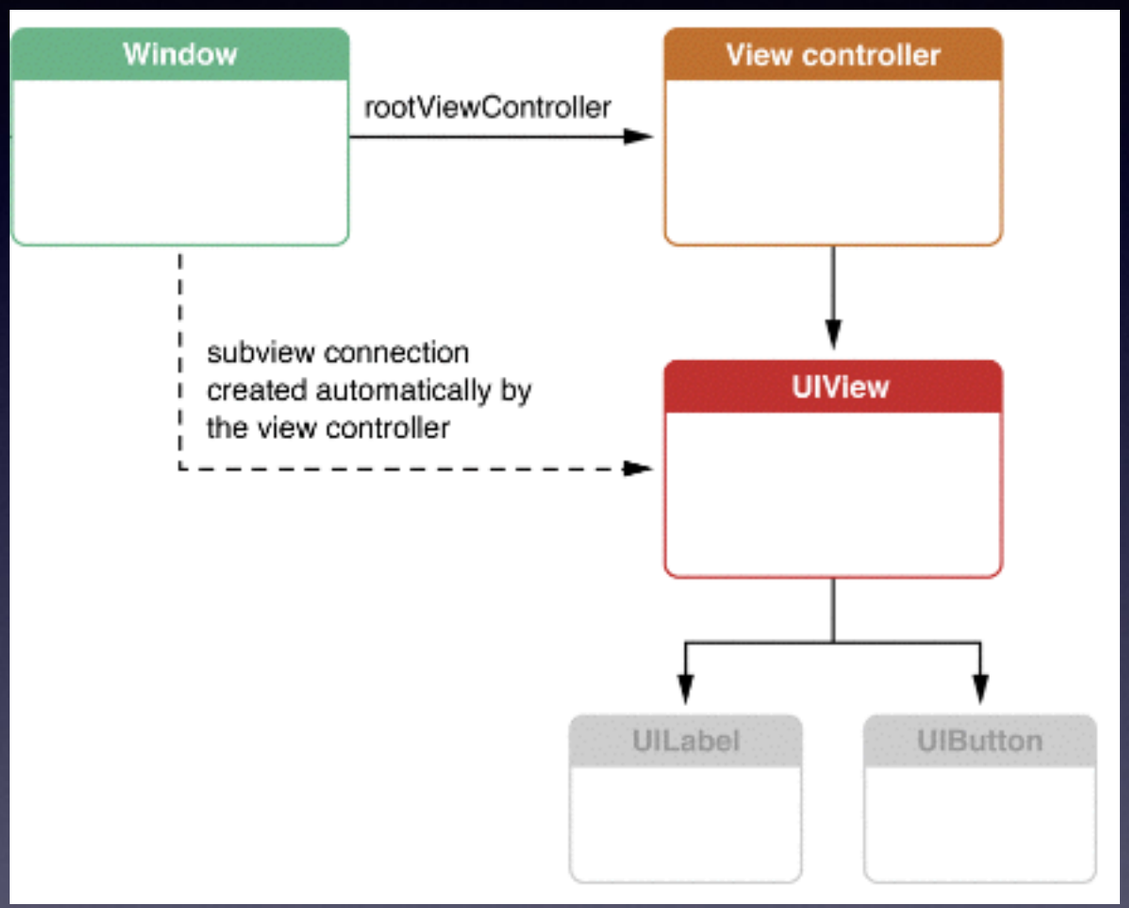
- brightness - 介於 0~1 的浮點數，用來設定目前螢幕的明暗度
- wantsSoftwareDimming - 設定是否允許使用軟體的方式來將畫面調整的比實際硬體允許的亮度還要暗
- 調整過度掃描
  - overscanCompensation - 用來補償過度掃描的參數

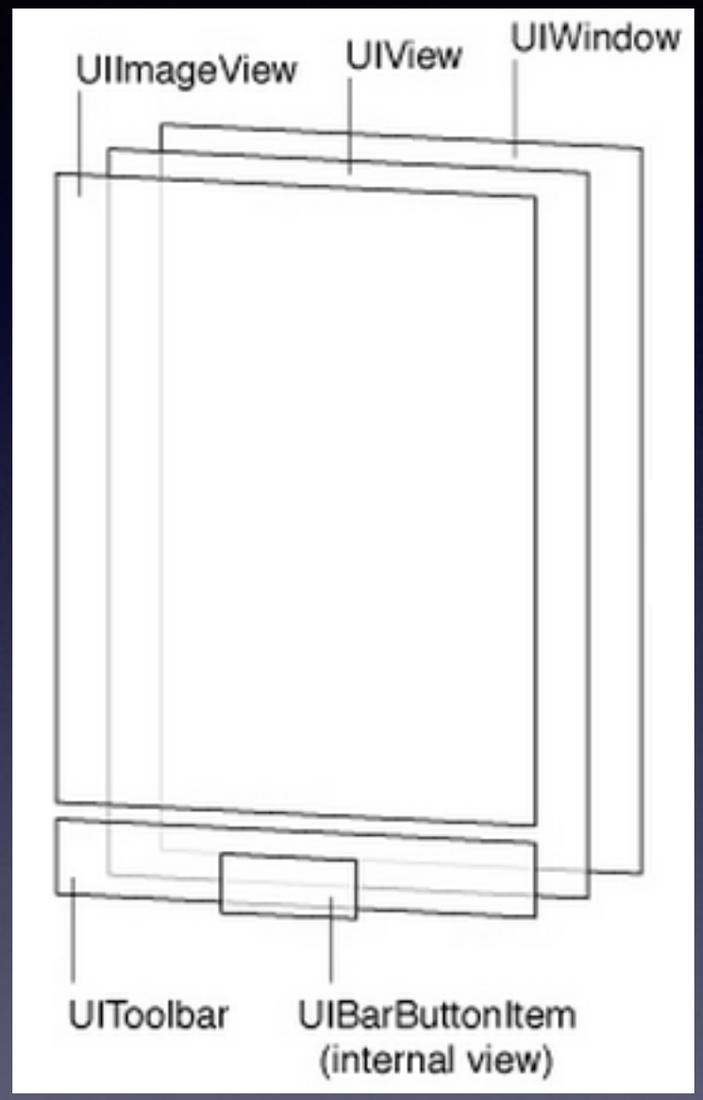
- UIWindow
  - UIResponser - 處理各種事件的類別，ex. 多點觸碰事件、搖晃事件
  - UIView - 放置各種視覺化元件的容器

- 在 iOS 每一個應用程式可以有一到多個 UIWindow (大部份只有一個)
- 每一個 UIWindow 裡可以有一到多個 UIView

- UIWindow - 視窗
- UIView - 視窗內的畫面元件
- 在 iOS 應用程式裡，元件的關係就好比圖層的關係，所以 UIWindow 才可以把接收到的事件給每一個畫面


NSObject
UIResponder
UIView
UIWindow
UILabel
UIPickerView
UIProgressView
UIActivityIndicatorView
UIImageView
UITabBar
UIToolbar
UINavigationBar
UITableViewCell
UIActionSheet
UIAlertView
UIScrollView
UITableView
UITextView
UISearchBar
UIWebView
UIControl
UIButton
UIDatePicker
UIPageControl
UISegmentedControl
UITextField
UISlider
UISwitch


Window
rootViewController
View controller
subview connection
created automatically by
the view controller
UIView
UILabel
UIButton


UIImageView
UIView
UIWindow
UIToolbar
UIBarButtonItem
(internal view)

代理機制與事件處理

- Delegate 必須遵循 UIApplicationDelegate 協定 (Protocol)
- 可以協助處理一些與應用程式有關的事件

- 應用程式生命週期相關事件
  - - (void)applicationDidFinishLaunching: (UIApplication *)application
    - 應用程式完成啟動後被呼叫
    - 可以做初始化動作
    - 或是恢復應用程式在上一次的狀態
    - 發送 UIApplicationDidFinishLaunchingNotification 訊息

- - (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:(NSDictionary *)launchOptions
  - 如果同時實作上述兩個方法，有 launchOptions 的這個方法將會被呼叫
- launchOptions 參數包含了應用程式被啟動時的資訊
  - 通知啟動 - launchOptions 包含 Badge Number、警告聲音、警告訊息或是其他使用者自訂資料，資料格式以 JSON 格式封裝再轉為 NSDictionary 物件傳進來

- URL 啟動 - launchOptions 包含 URL 及應用程式 ID
- 使用者點擊啟動 - launchOptions 為 nil

- - (void)applicationWillResignActive:(UIApplication *)application
  - 當應用程式由睡眠狀態進入啟動狀態時被呼叫
  - 會發出 UIApplicationWillResignActiveNotification 訊息

- - (void)applicationWillTerminate:(UIApplication *)application
  - 當應用程式將被終止執行時被呼叫
  - 通常清除使用的資源
  - 會發出 UIApplicationWillTerminateNotification 訊息
- - (void)applicationDidReceiveMemoryWarning:(UIApplication *)application

- 當記憶體不足時被呼叫
- 有使用緩衝的機制最好在這個方法內清除原有的緩衝資料
- 會發出 UIApplicationDidReceiveMemoryWarningNotification 訊息
- - (void)applicationSignificantTimeChange:(UIApplication *)application
  - 當子夜來臨時，來自電信營運商的時間更新或是日光節約時間改變時被呼叫

- 會發出 UIApplicationSignificantTimeChangeNotification 訊息

- - (void)applicationWillEnterForeground:(UIApplication *)application

  - iOS 4.0 以後的事件

  - 當應用程式進入前景時被呼叫

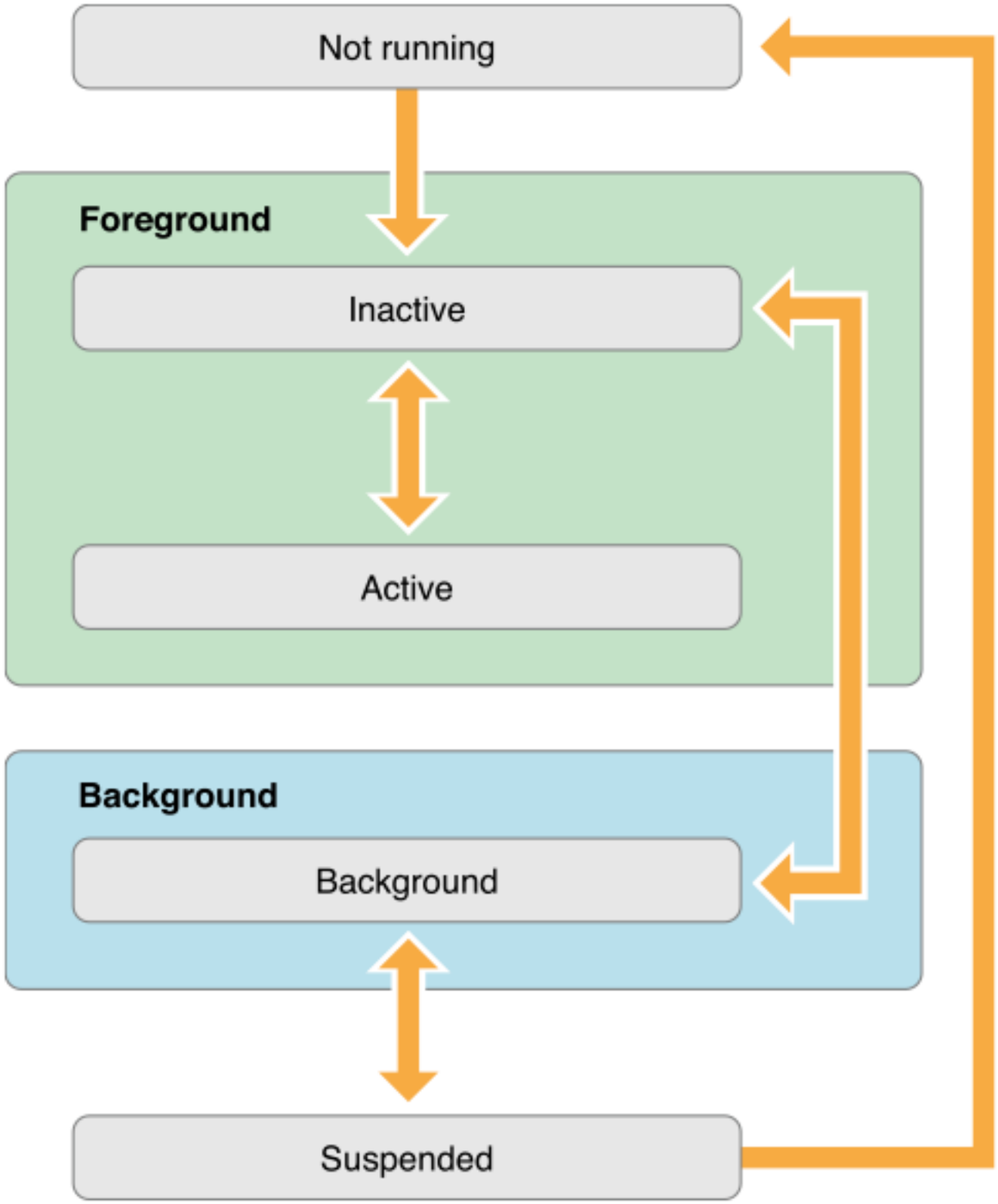
- - (void)applicationDidEnterBackground:(UIApplication *)application

  - iOS 4.0 以後的事件

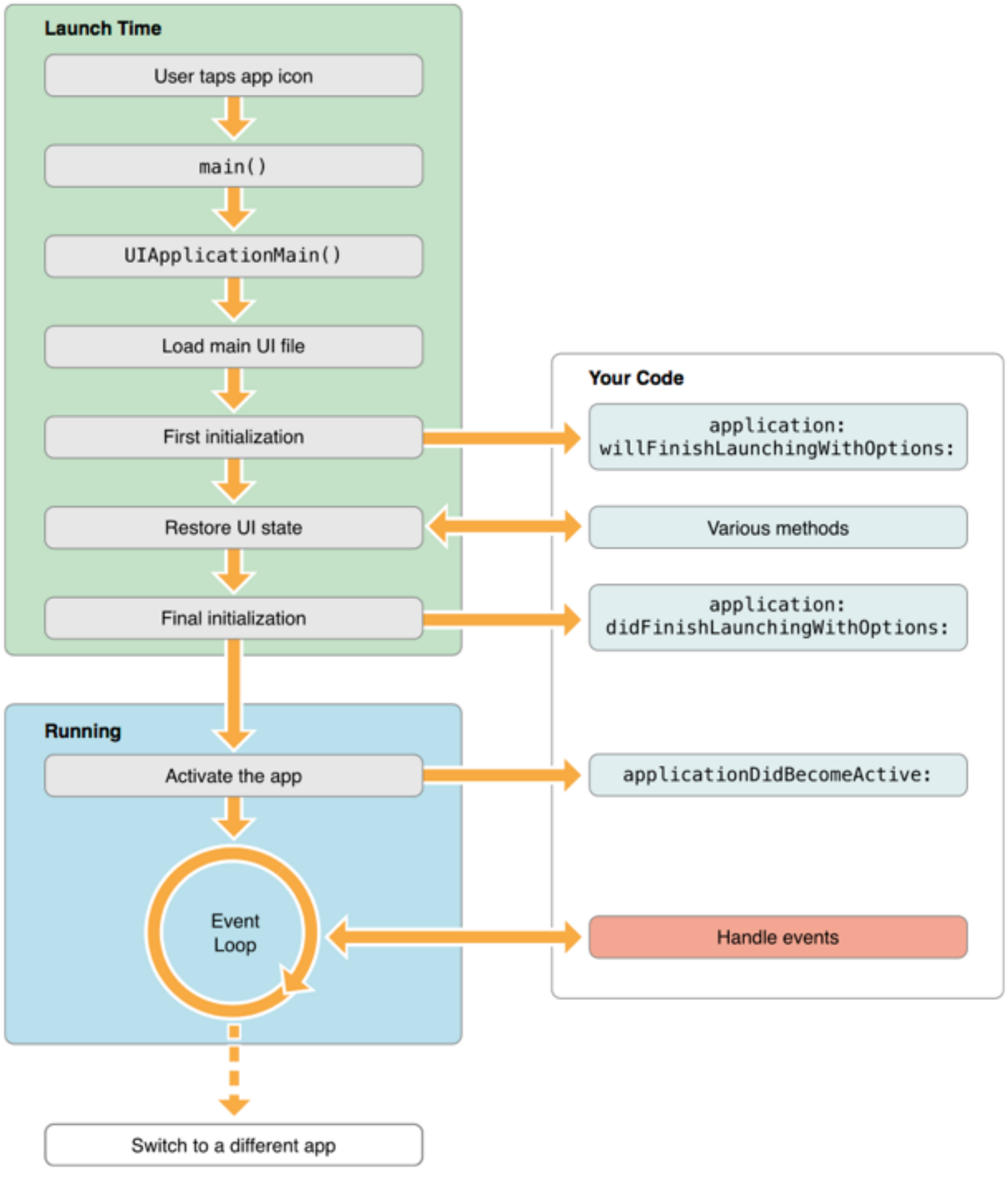
  - 當應用程式進入背景時被呼叫

- 應用程式啟動時
- 應用程式移至背景時
- 應用程式被中斷時

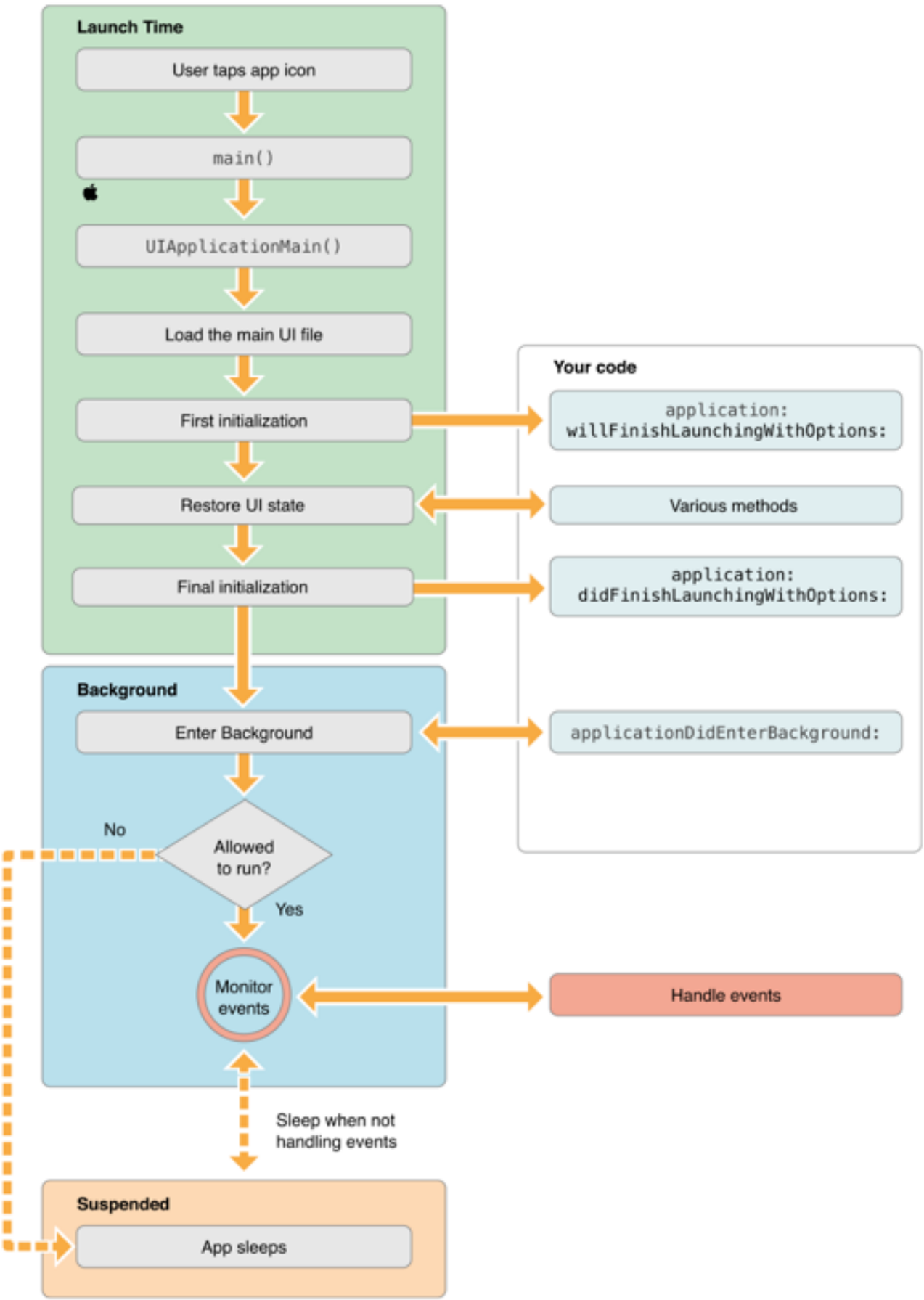
State changes in an iOS app



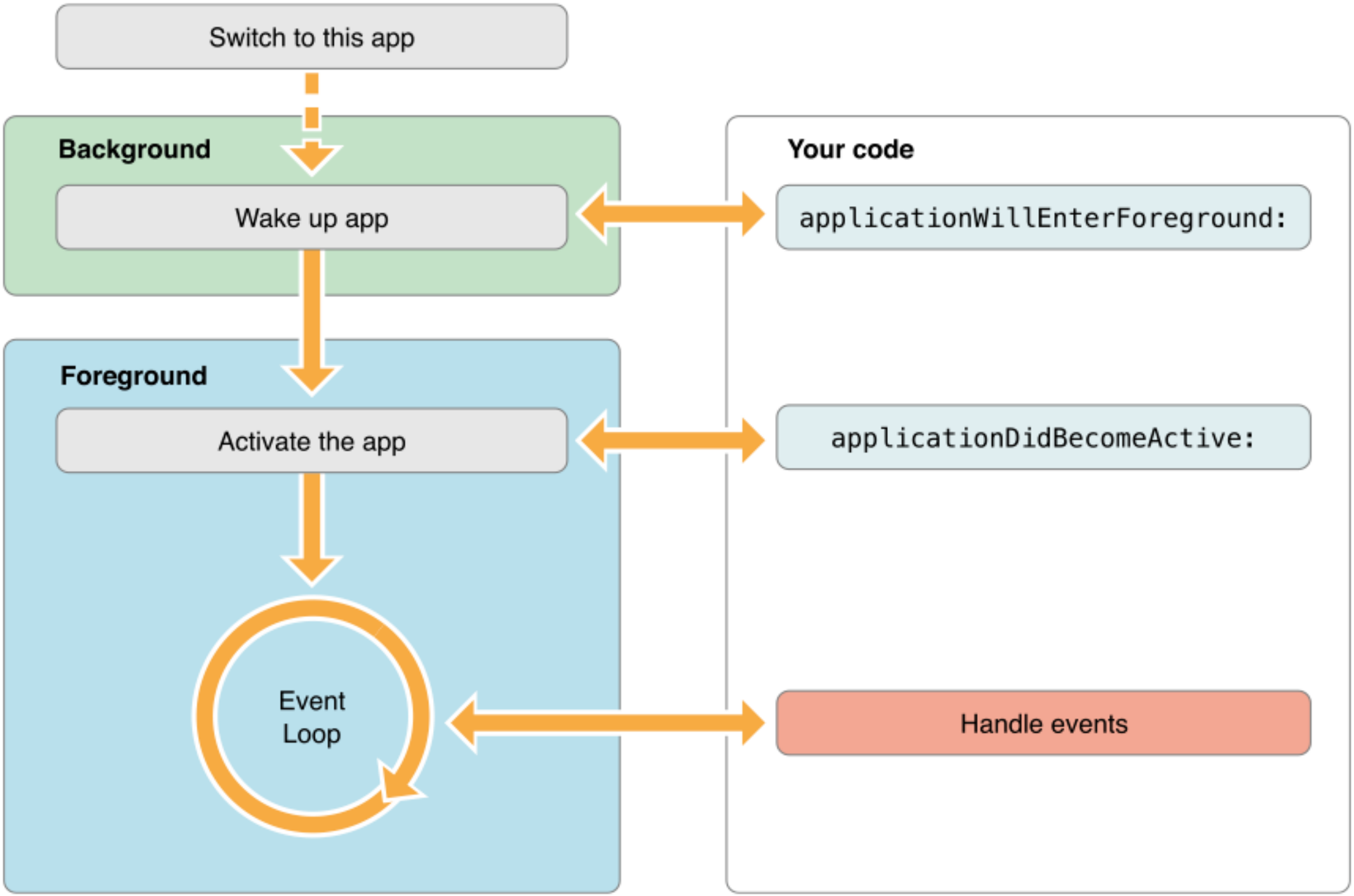
Launching an app into the foreground

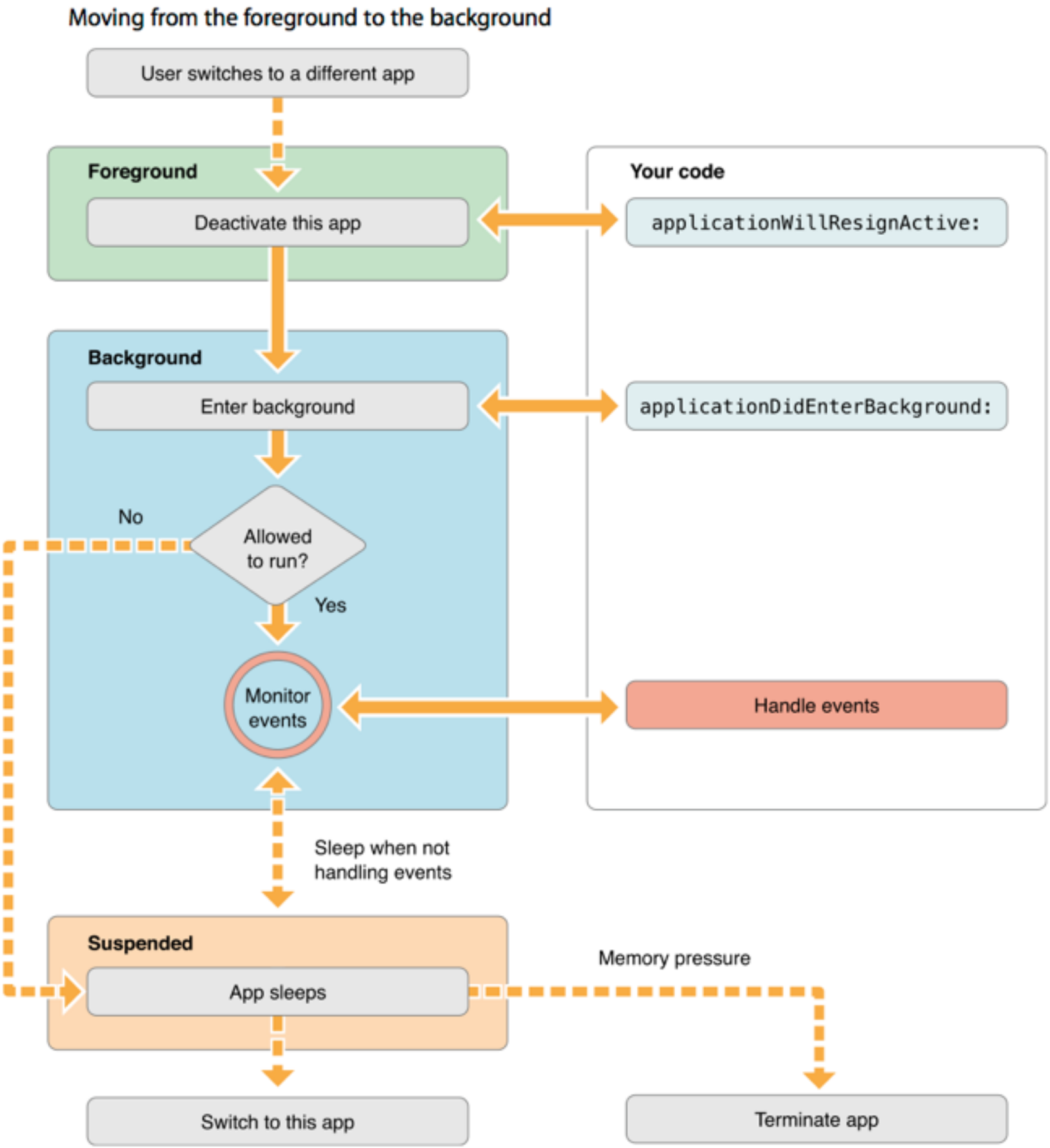


Launching an app into the background



Transitioning from the background to the foreground




Moving from the foreground to the background
User switches to a different app
Foreground
Deactivate this app
Your code
applicationWillResignActive:
Background
Enter background
applicationDidEnterBackground:
No
Allowed to run?
Yes
Monitor events
Handle events
Sleep when not handling events
Suspended
App sleeps
Memory pressure
Switch to this app
Terminate app

- 狀態列的事件
  - - (void)application:(UIApplication *)application willChangeStatusBarOrientation:(UIInterfaceOrientation)newStatusBarOrientation duration:(NSTimeInterval)duration
  - 當狀態列的方向將要被改變時被呼叫

- - (void)application:(UIApplication *)application didChangeStatusBarOrientation:(UIInterfaceOrientation)oldStatusBarOrientation
- 當狀態列的方向被改變之後會被呼叫
- 會發出 UIApplicationDidChangeStatusBarOrientationNotification 訊息

- - (void)application:(UIApplication *)application willChangeStatusBarFrame:(CGRect)newStatusBarFrame
  - 當狀態列的邊框區域改變時會被呼叫
- - (void)application:(UIApplication *)application didChangeStatusBarFrame:(CGRect)oldStatusBarFrame
  - 當狀態列的邊框區域改變後會被呼叫

訊息通知機制

- 通知 (Notification) 機制是採用廣播的方式
- 將要發送出去的訊息廣播給註冊過的觀測者 (Observer) 知道，由觀測者對訊息進行處理
- 通知機制像是廣播站一樣，可以讓很多的物件都接收到所發送出來的訊息

- 代理機制只有接收代理的物件會接收到訊息
- 觀測者可以接收到任意數量的訊息通知
- 代理機制只能將訊息發送給預定好的某個方法
- 發送通知者只需要負責將訊息發送出去就好

- 透過觀測者接收訊息
  - NSNotificationCenter 是這個機制運行的核心類別
  - 應用程式裡的每一個進程 (Process) 都會有一個獨一無二的 NSNotificationCenter 的實例存在
- 所發送的訊息以 NSNotification 類別型態存在
- NSNotificationCenter *center = [NSNotificationCenter defaultCenter];

- 要讓某個物件可以監聽訊息中心所發送出來的訊息必須加入一個觀測者

- - (void)addObserver:(id)notificationObserver selector:(SEL)notificationSelector name:(NSString *)notificationName object:(id)notificationSender

- [center addObserver:myObserver selector:@selector(receiveNotification) name:@“ValueChange” object:self];

- NSDictionary *userInfo = [[NSDictionary alloc] initWithObjectsAndKeys:@“1st Value”, @“key1”, @“2nd Value”, @“key2”, nil];
- 也必須適時的移除掉 [center removeObserver:myObserver];

Target-Action 機制

- 元件都與多點觸控事件有關
- 在 UIControl 定義 Target-Action 機制
- target 泛指代表這些元件的物件
- action 是當事件發生後被定義在 target 內會被呼叫的方法

- UIControl 類別的元件透過 addTarget:action:forControlEvents: 方法設定好事件以及處理的方法
- 在 UIKit framework 內會建立一張表格記錄每個事件以及每對 Target-Action 的對應關係
- 當使用者觸碰螢幕上的 UIControl 元件會發出 sendActionsForControlEvents: 訊息

- 訊息的發送會導致 UIApplication 的 sendAction:to:from:forEvent 被呼叫
- 由 UIApplication 集中處理後再呼叫相對應對象 (Target) 的方法 (Action)