ITM103 iOS Application Development

Topic 9: App States and Multitasking

Objectives

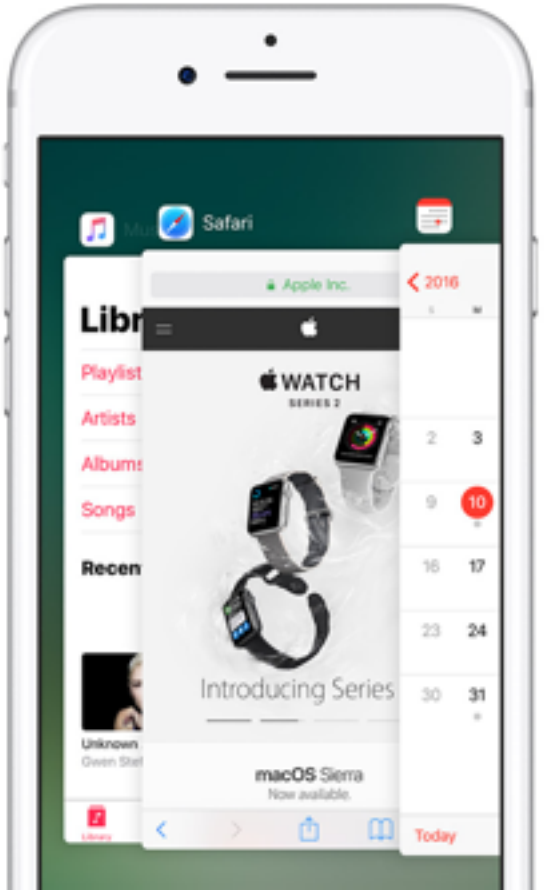
- By the end of the lesson, you will be able to:
  - Examine the different application states
  - Explain App Life Cycle
  - Handle app state transitions
  - Opt Out of the Background Mode
  - Detect Multi-tasking
  - know how to save, load application states using archiving/unarchiving

App States and Multitasking

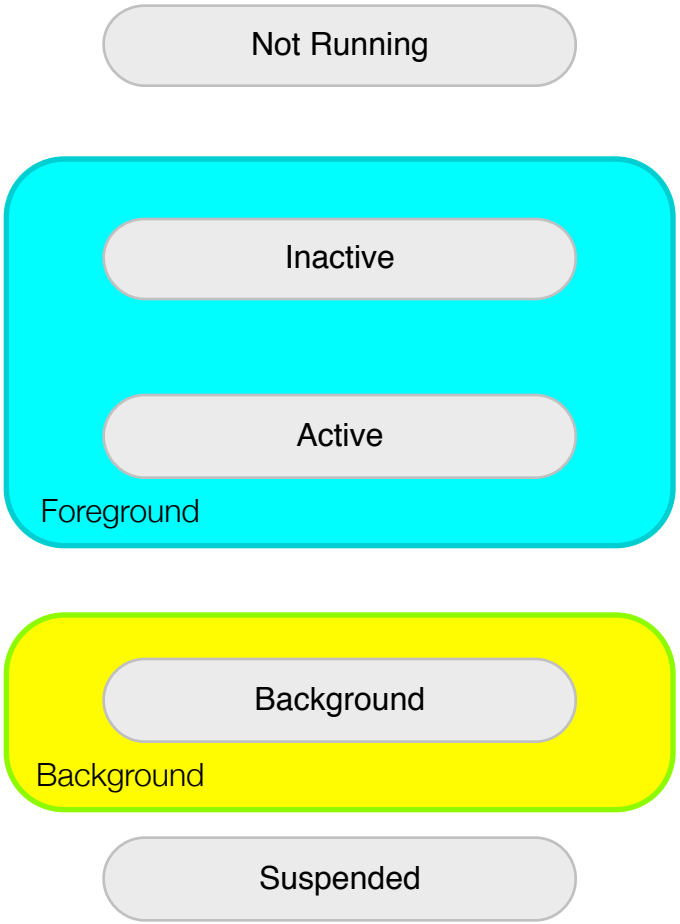
Application States

App States and Multitasking

- System resources are limited.
- Important to know if the app is running in the foreground or the background.
- Different behaviour when running in foreground / background.

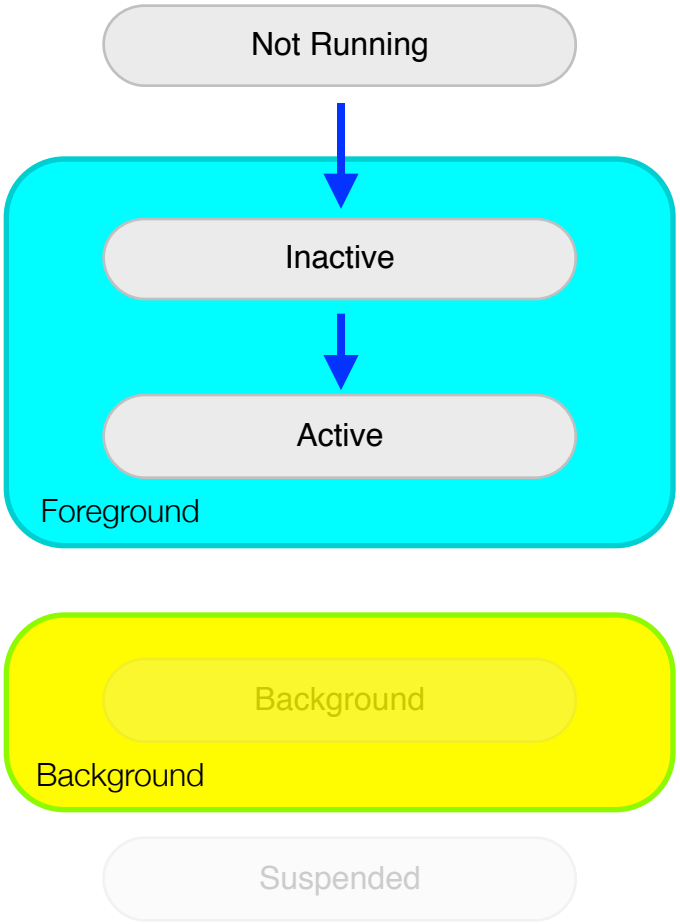


States in an iOS App



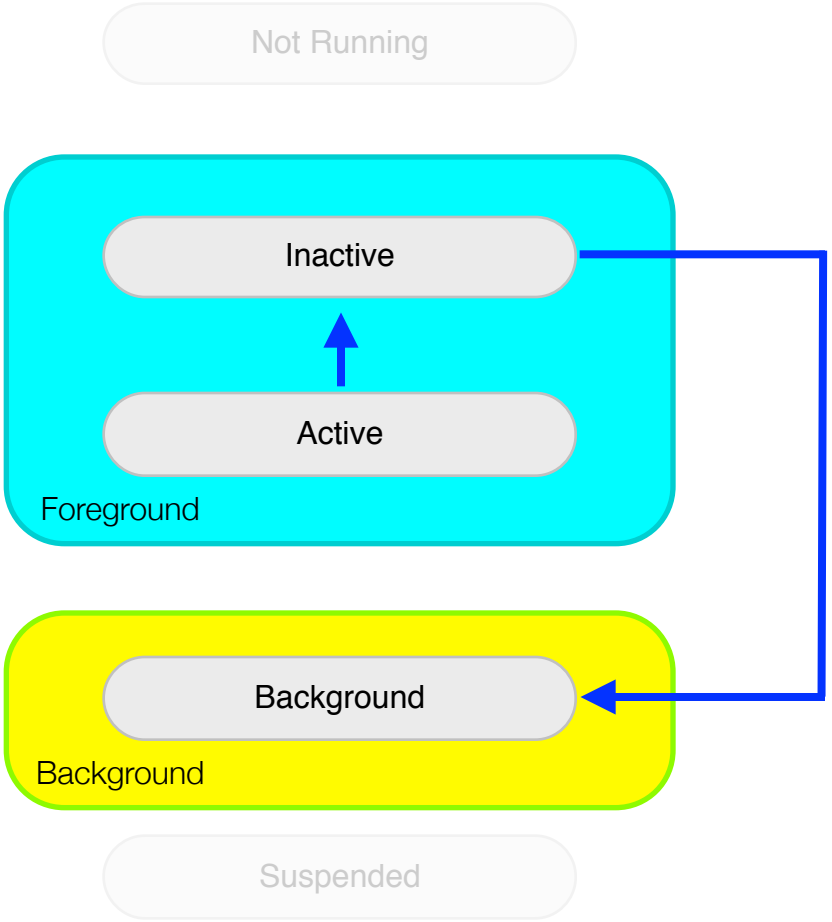
State Changes in an iOS App

When the user newly opens up an app.



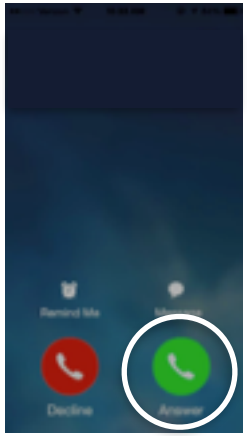
State Changes in an iOS App

User presses home button and moves your app into the background



OR

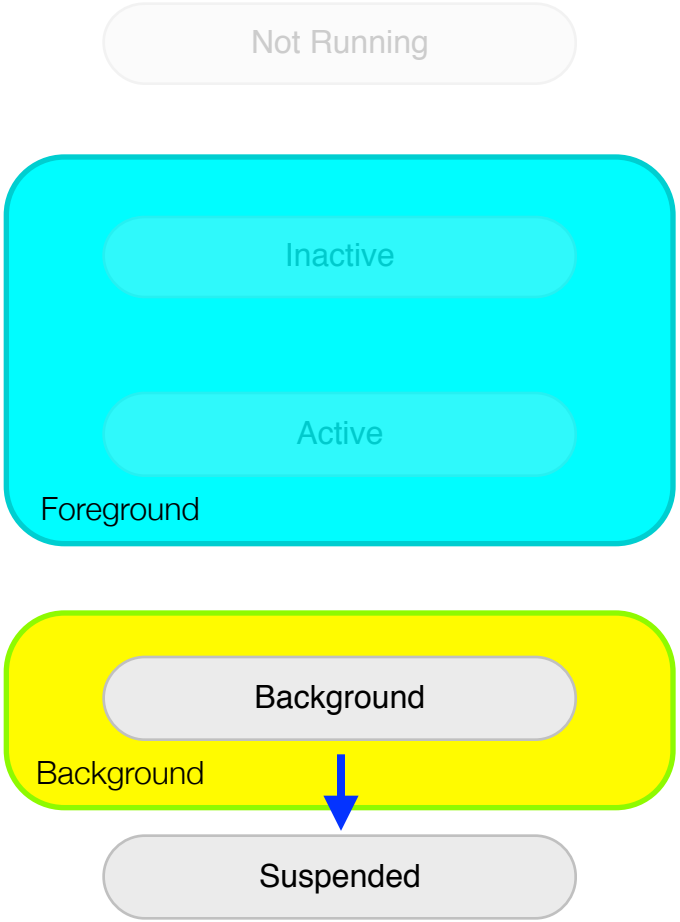
Phone call comes in, user picks up.



State Changes in an iOS App

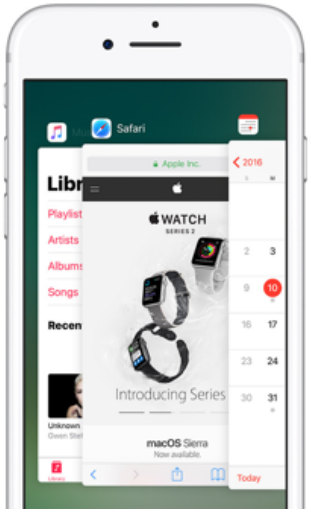
Most apps don't stay in background too long.

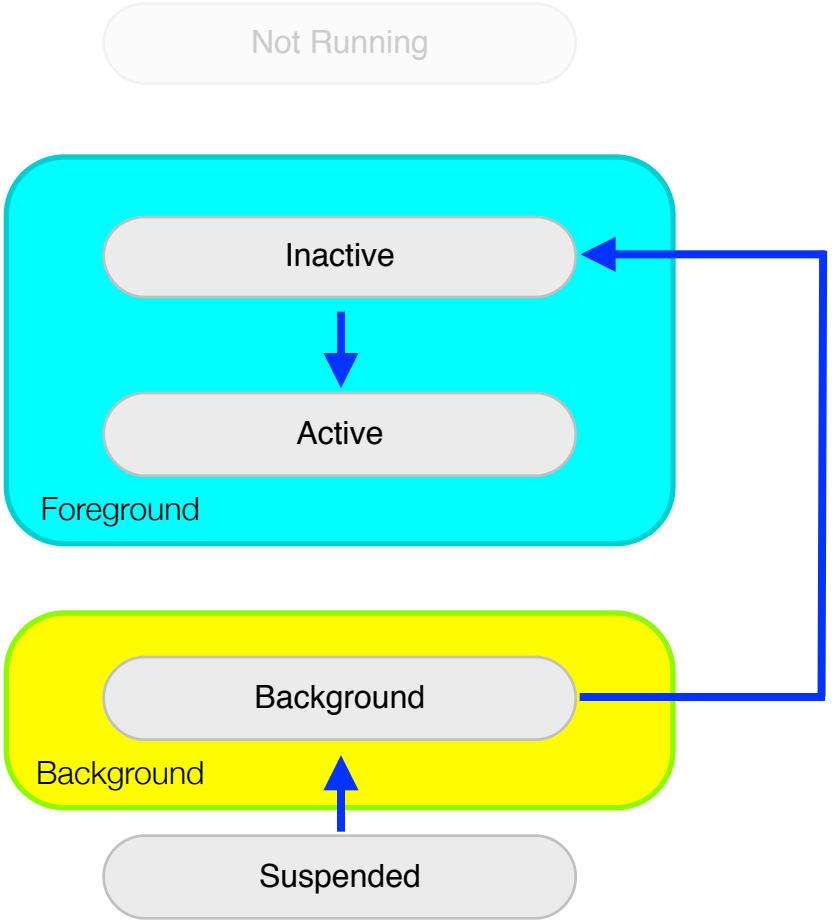
iOS will move them into a **Suspended** state



State Changes in an iOS App

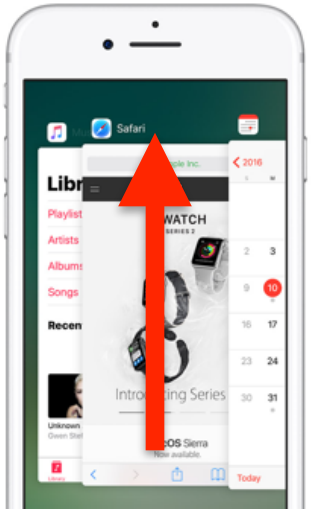
User re-opens your app.

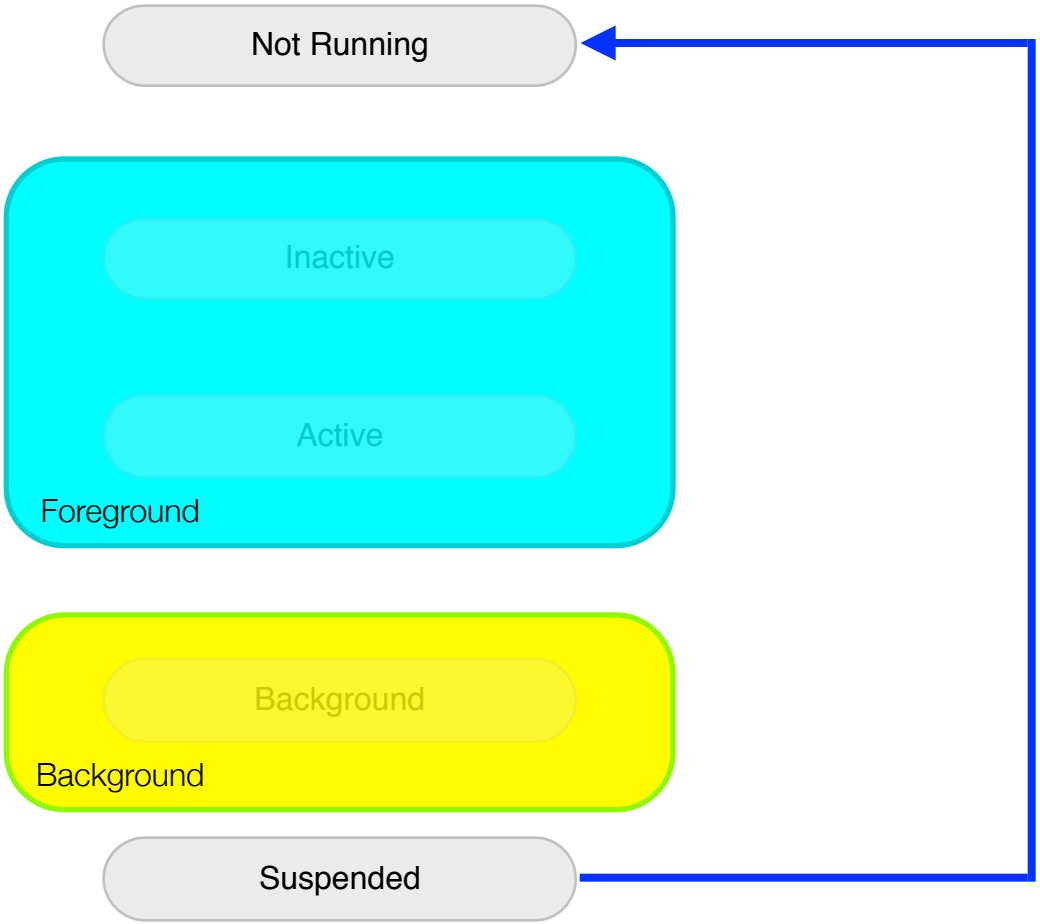




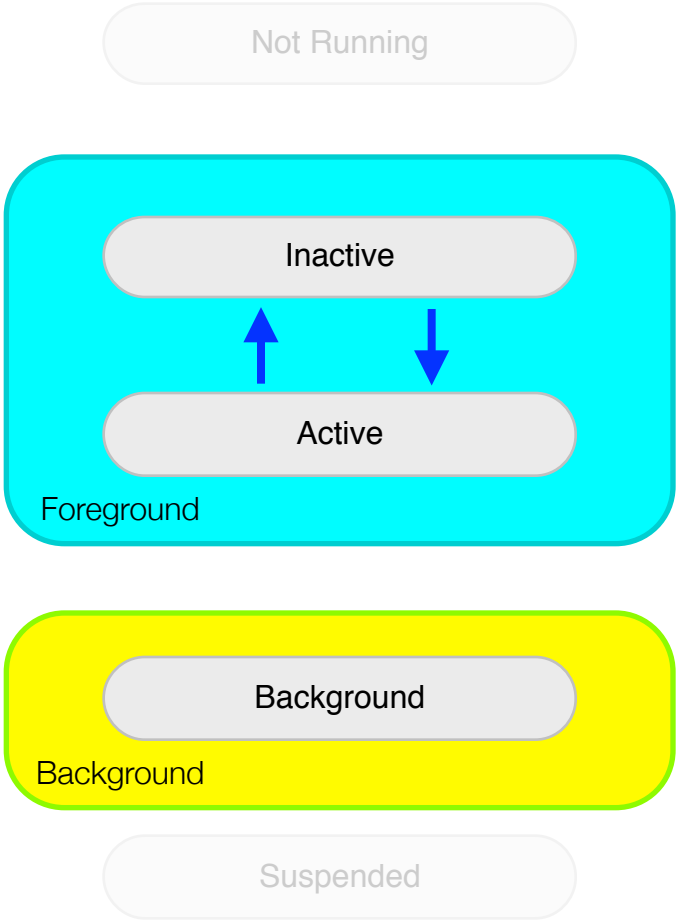
State Changes in an iOS App

The user (swipe up) or iOS terminates the app.

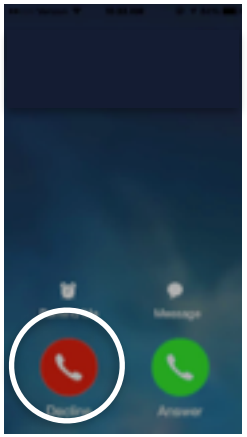




State Changes in an iOS App

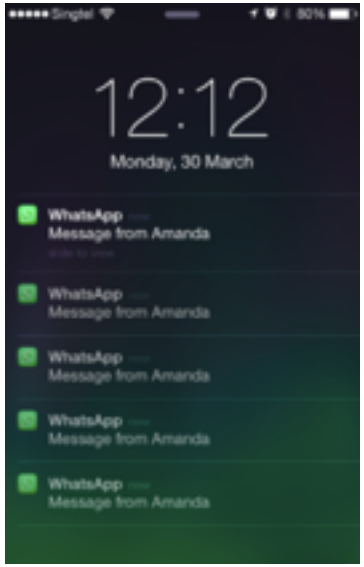


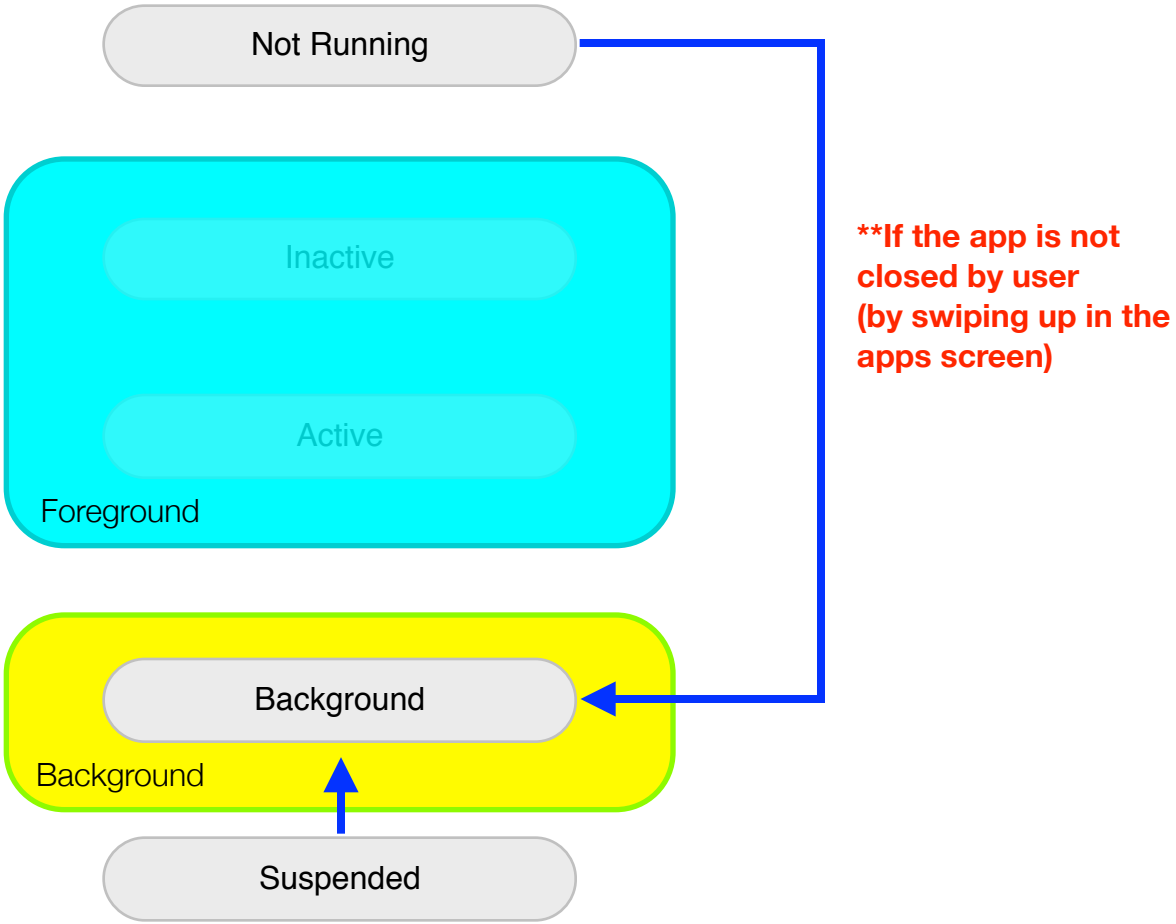
Phone call comes in, user cancels.



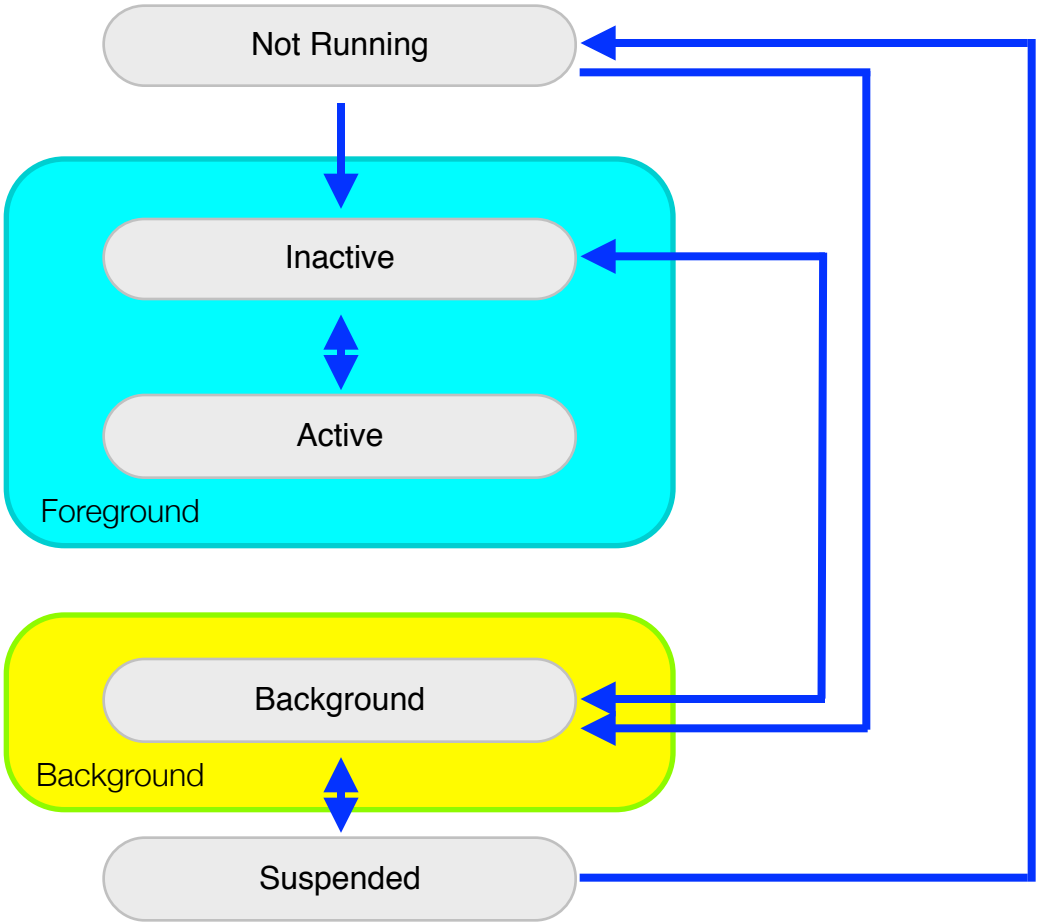
State Changes in an iOS App

When your app receives a Push notification, or performs a Background Fetch





State Changes in an iOS App



Reference: https://developer.apple.com/library/ios/documentation/iphone/conceptual/iPhoneOSProgrammingGuide/TheAppLifeCycle/TheAppLifeCycle.html#//apple_ref/doc/uid/TP40007072-CH2-SW1

Summary of App States

State	Description
Not running	The app has not been launched or was running but was terminated by the system.
Inactive	The app is running in the foreground but is currently not receiving events. (It may be executing other code though.) An app usually stays in this state only briefly as it transitions to a different state.
Active	The app is running in the foreground and is receiving events. This is the normal mode for foreground apps.
Background	The app is in the background and executing code. Most apps enter this state briefly on their way to being suspended. However, an app that requests extra execution time may remain in this state for a period of time. In addition, an app being launched directly into the background enters this state instead of the inactive state. For information about how to execute code while in the background, see "Background Execution and Multitasking."
Suspended	The app is in the background but is 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 any code. When a low-memory condition occurs, the system may purge suspended apps without notice to make more space for the foreground app.

Reference: https://developer.apple.com/library/ios/documentation/iphone/conceptual/iPhoneOSProgrammingGuide/TheAppLifeCycle/TheAppLifeCycle.html#//apple_ref/doc/uid/TP40007072-CH2-SW1

App States and Multitasking

Application Events Summary

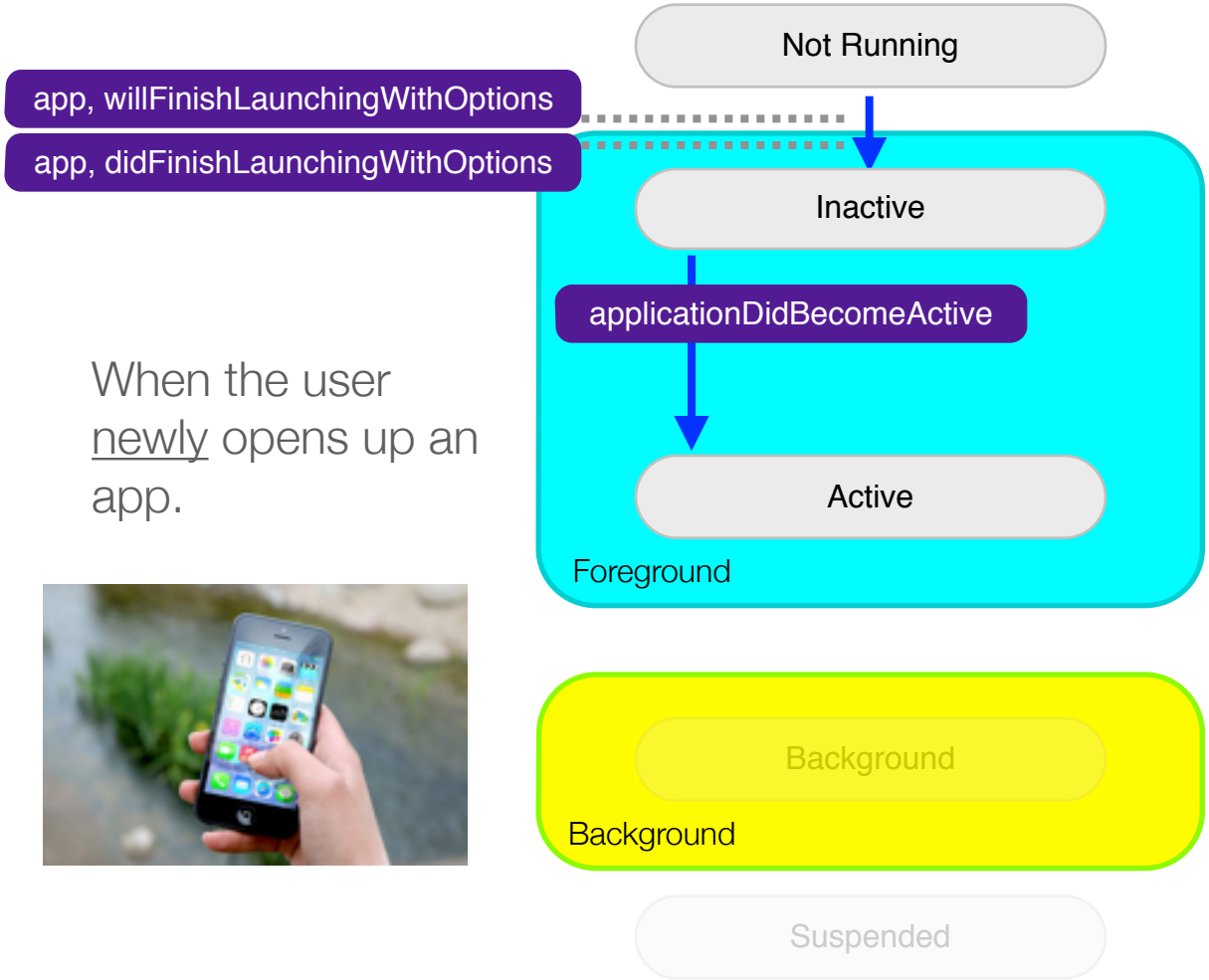
App Events

- Like View Controllers, you can implement your own logic at specific Application Events.

- You can do that in the

AppDelegate.swift

App Events During State Changes



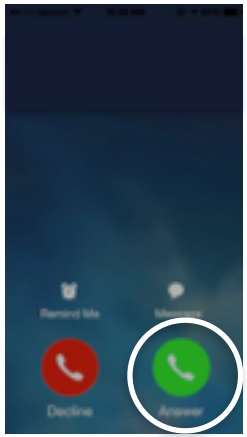
When the user newly opens up an app.

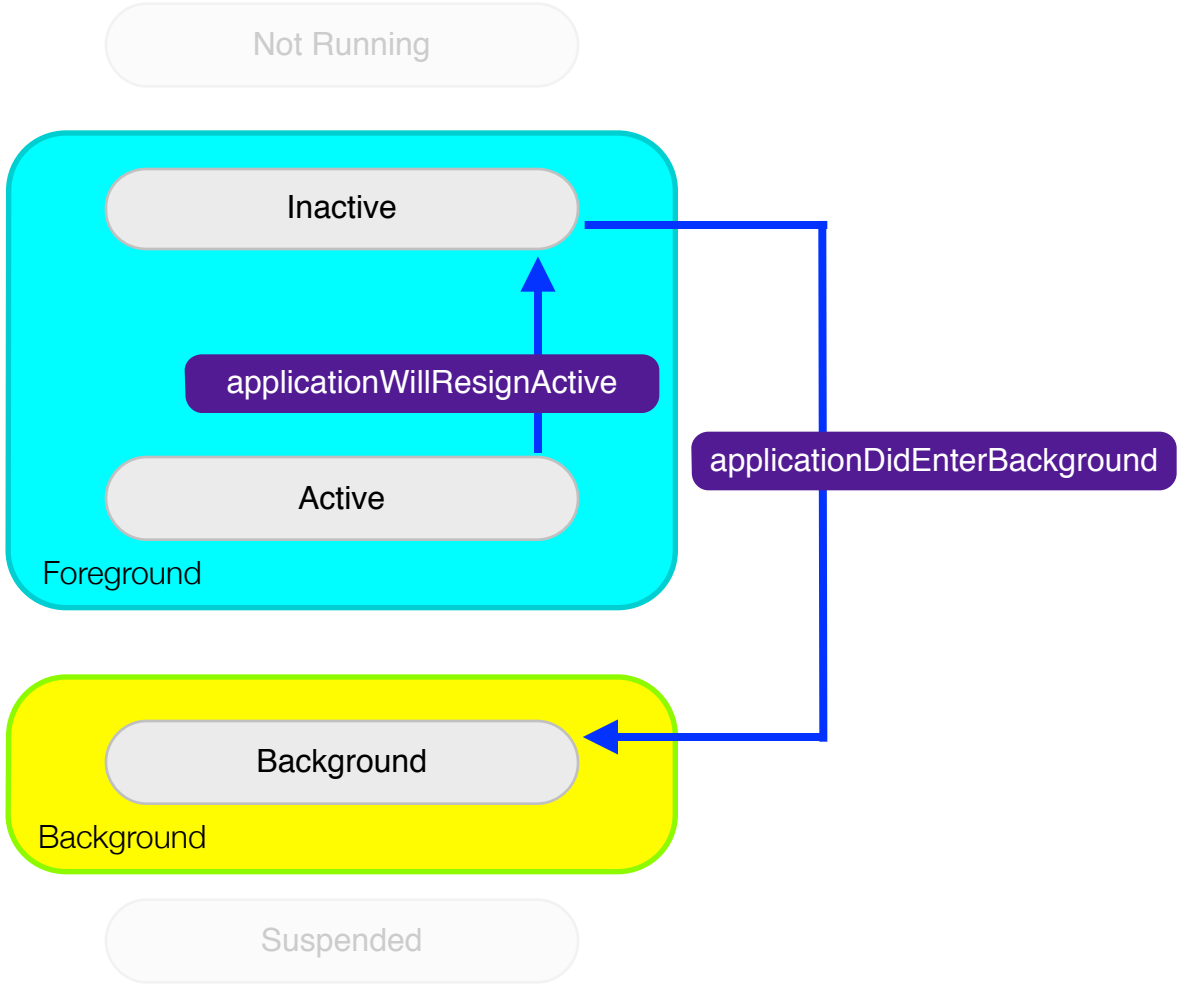
App Events During State Changes

User presses home button.

or

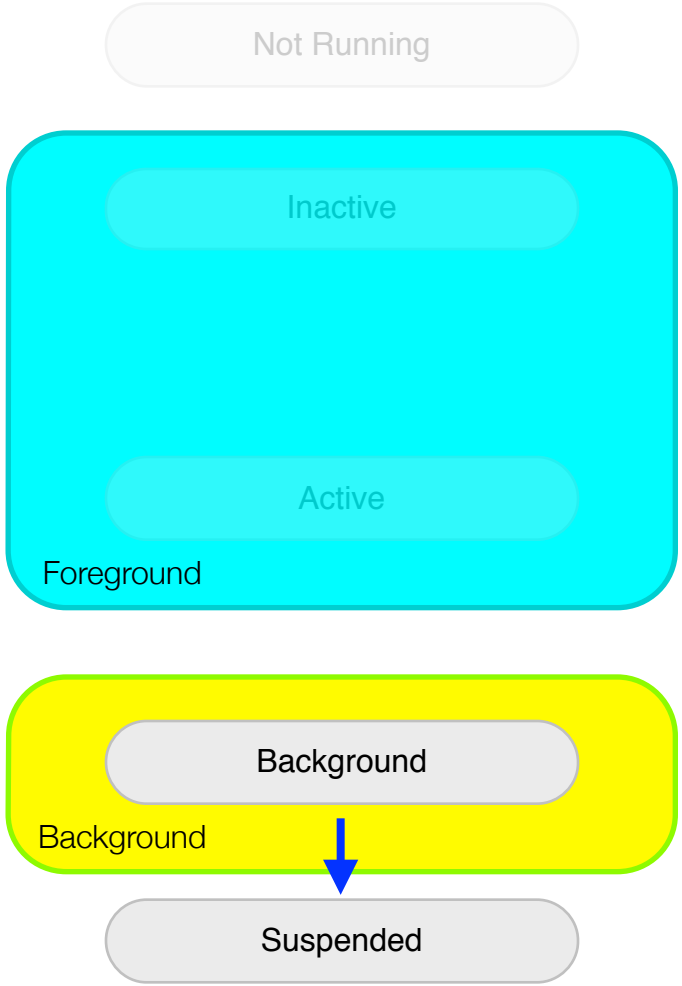
Phone call comes in, user picks up.





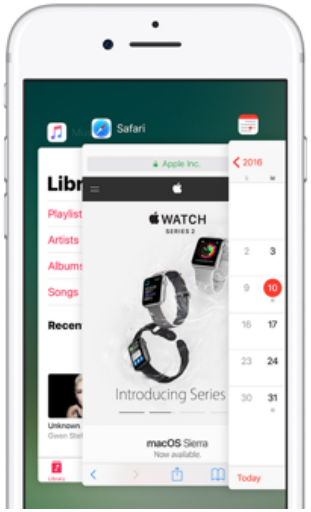
App Events During State Changes

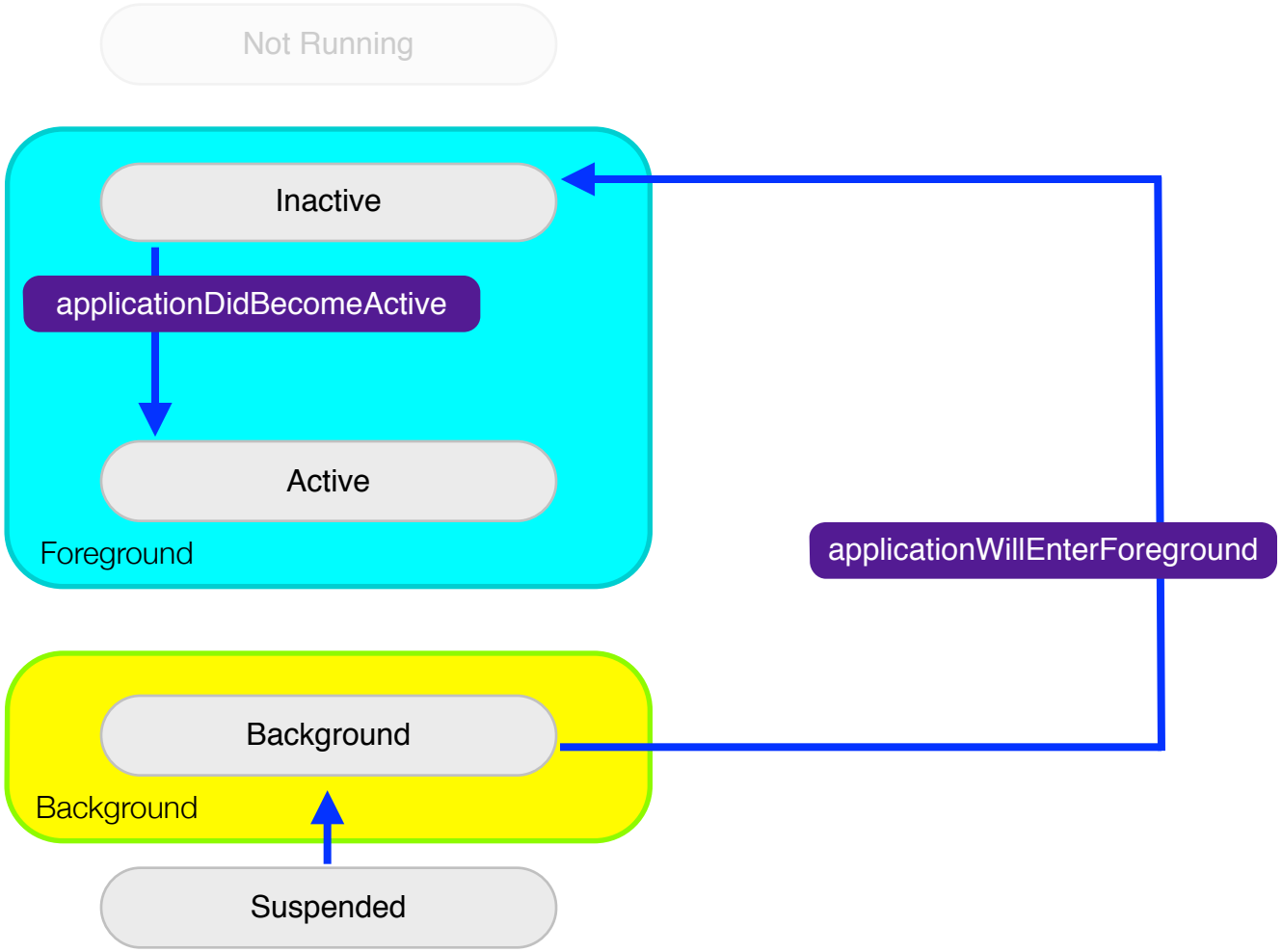
iOS suspends your app.



App Events During State Changes

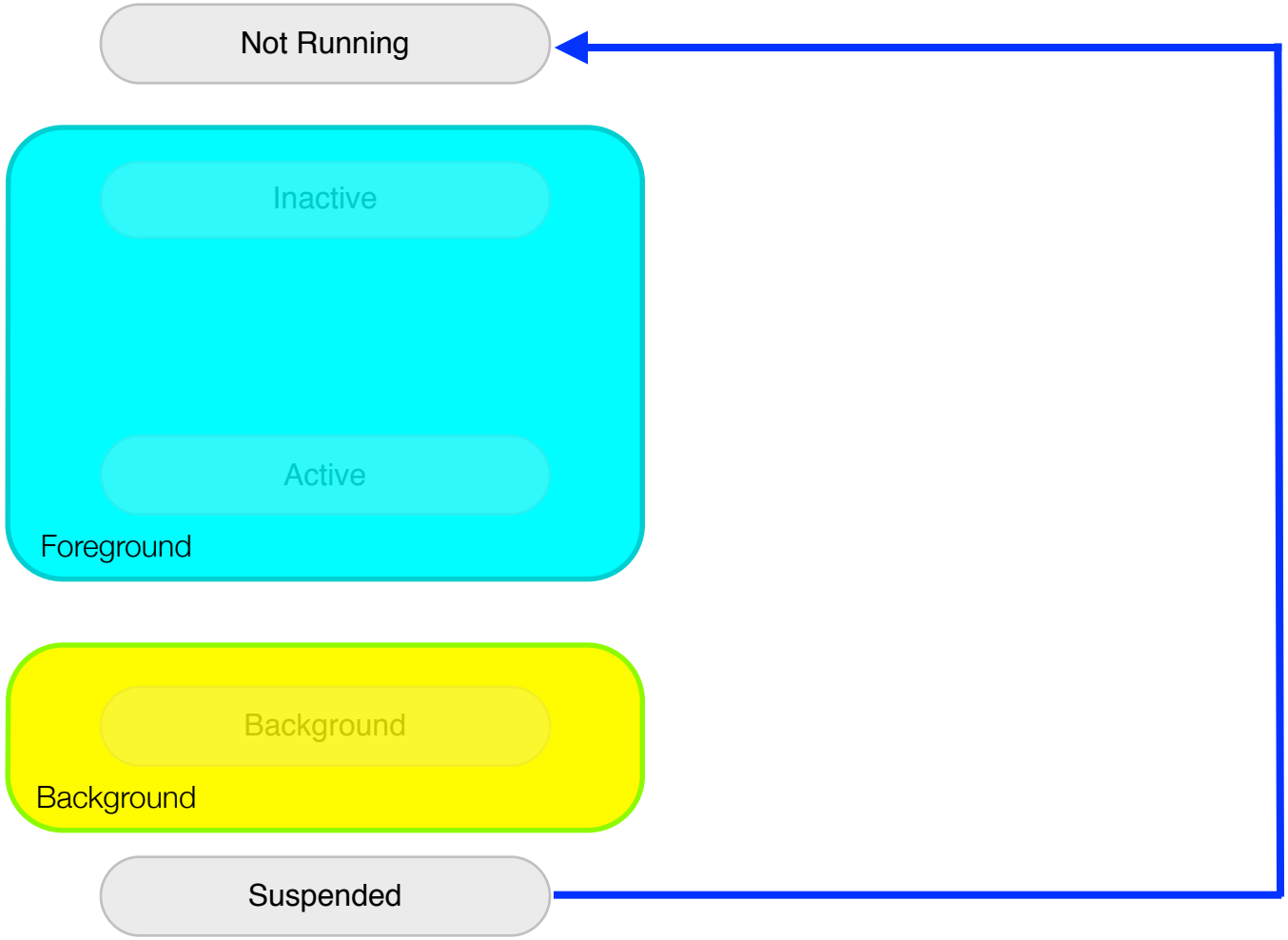
User re-opens your app.





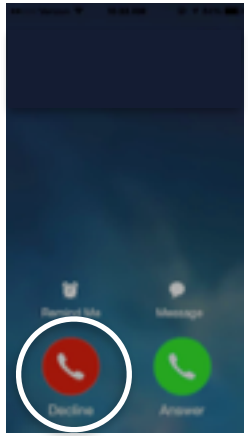
App Events During State Changes

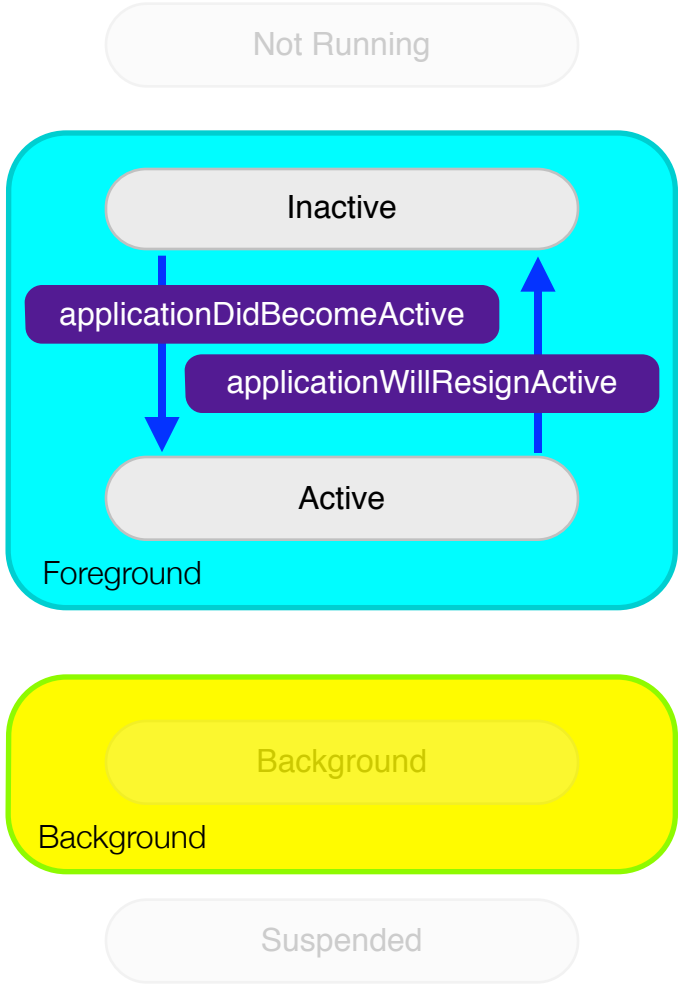
User or iOS terminates your app.



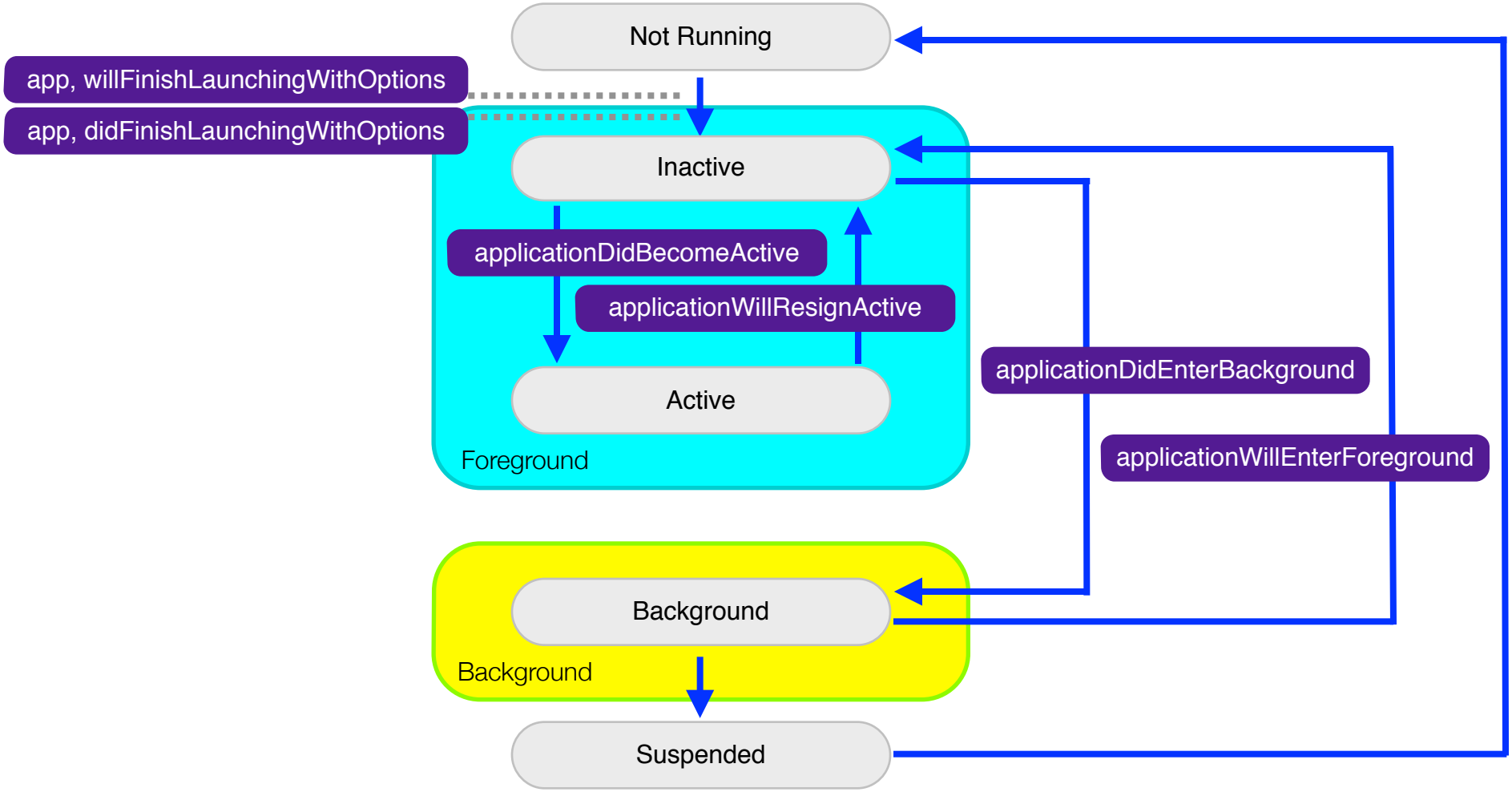
App Events During State Changes

Phone call comes in, but user cancels.





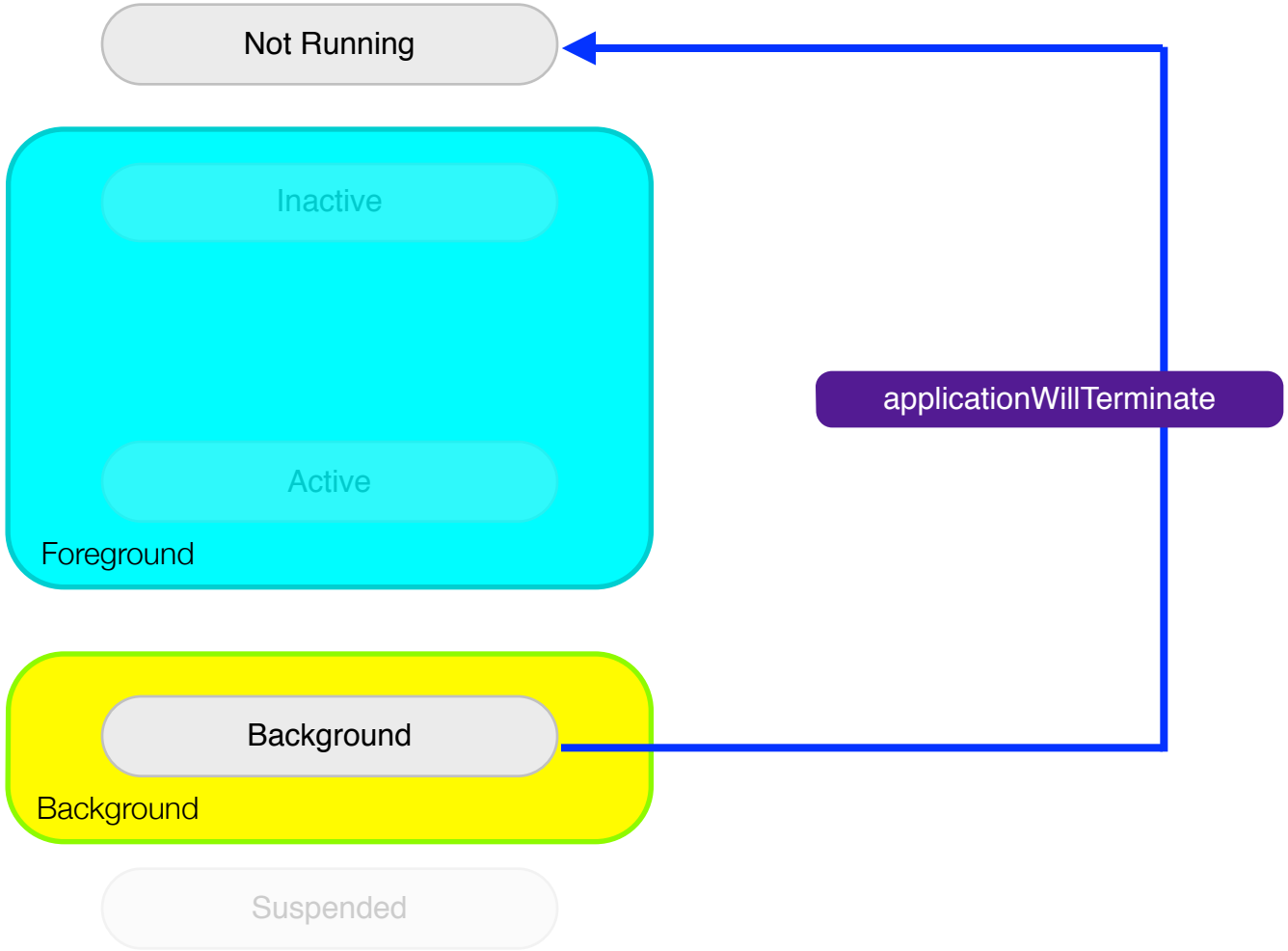
App Events During State Changes



App Termination from Background

Special case

Your app is running in the Background, but iOS terminates it.

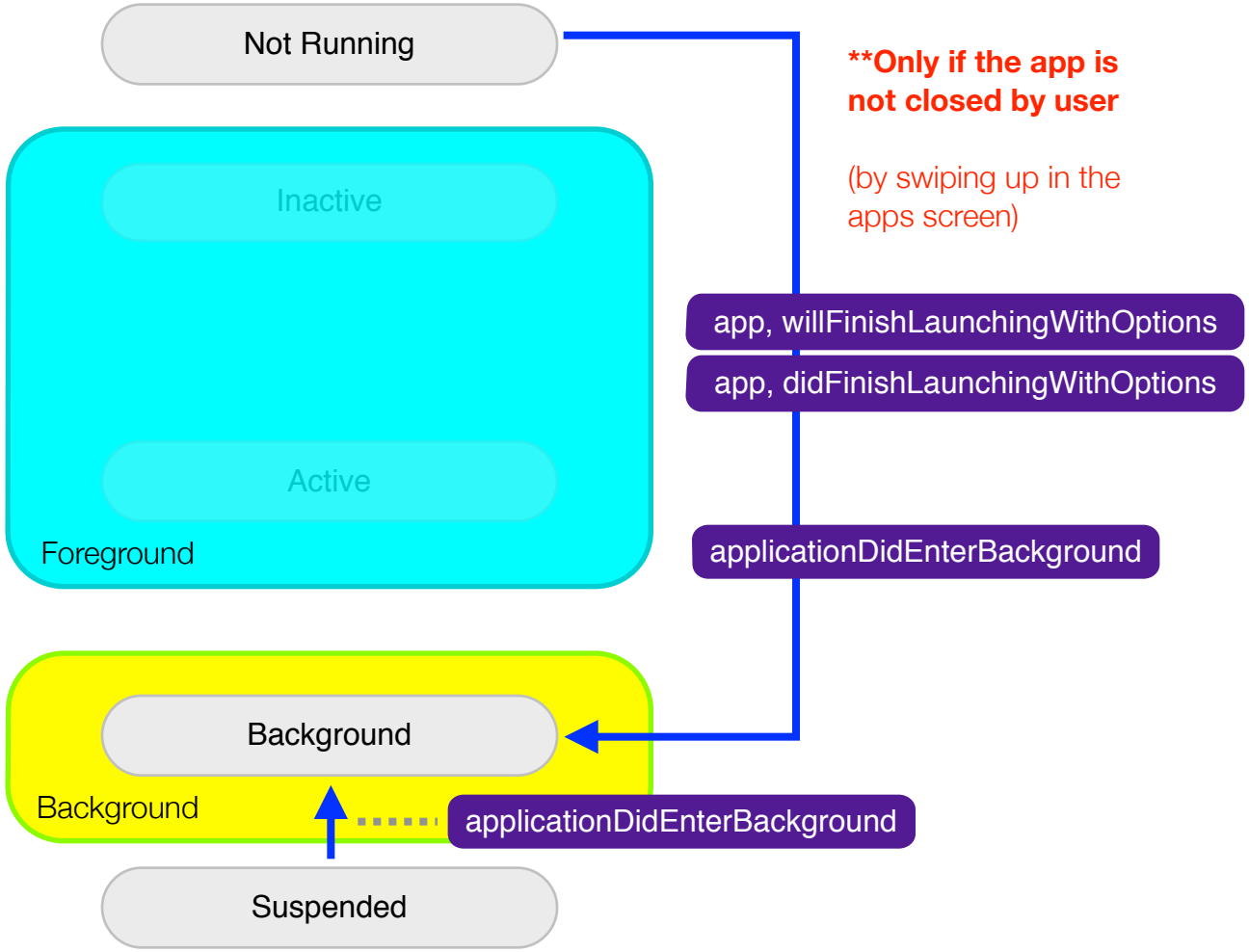


App Launching into Background

The user taps on a Push Notification and launches your app,

or

iOS triggers your app to perform a Background App Refresh



Summary of App Events

Methods	Description
application(application, willFinishLaunchingWithOptions)	First executed method when the app is launched.
application(application, didFinishLaunchingWithOptions)	This method allows you to perform your final initialization before your app is displayed to the user.
applicationDidBecomeActive	Is received after application(application, didFinishLaunchingWithOptions). Let your app know that it is about to enter to the foreground.
applicationWillResignActive	The most common cause is when the screen has been locked. Lets you know that your app is transitioning away from being the foreground app. Use this method to put your app into a quiescent state.

Summary of App Events

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

What to do at Launch Time

- When the app is launched:

```
application(application: willFinishLaunchingWithOptions:)
application(application: didFinishLaunchingWithOptions:)
```

  Use them to:

  – Check the contents of the launch options dictionary and respond appropriately.
  – Initialize the app's data structures
  – Prepare the app's windows and views for display.

Application States

- To determine whether your app is launching into the foreground or background, check

  `application.applicationState`

  in

  `application(application, willFinishLaunchingWithOptions)` or `application(application, didFinishLaunchingWithOptions)` delegate method.

Property Value	App State
UIApplicationState.Active	When the app is launched into the foreground by the user.
UIApplicationState.Background	When the app is launched into the background by iOS (due to Push Notifications, or Background Fetch)

What to do When an Interruption Occurs

- When the app moves out of Active state (due to incoming SMS / phone call / user quit):

  `applicationWillResignActive(application:)`

  Use it to:
  - Save data and any relevant state information
  - Stop timers and other periodic tasks.
  - Stop any running metadata queries
  - Do not initialize any new tasks.
  - Pause movie playback
  - Enter into a pause state if your app is a game
  - Suspend any dispatch queues

What to do When an Interruption Occurs

- When the app regains control:

```
func applicationDidBecomeActive(application:)
```

  Use it to:

  – Reverse steps in **`applicationWillResignActive`**

What to do When Moving to the Background

- When the app is moved into the background:

```
func applicationDidEnterBackground(application:)
```

  Use it to:

  - Save user data and app state information.
  - Invalidate timers.
  - Free up as much as memory as possible.
  - **Prepare to have their picture taken.**

What to do When Moving to the Background

Wait a minute

prepare to have our picture taken?

What to do When Moving to the Foreground

- When the user moves app into foreground:

```
func applicationWillEnterForeground(application:)
```

  Use it to:

  – Undo changes made when entering background.

What to do When Moving to the Foreground

- When the app regains control:

```
func applicationDidBecomeActive(application:)
```

  Use it to:

  – Restart tasks that were paused.
  – Restart timers.
  – Refresh user interface.

Determine if multitasking is available

- Check the

  `isMultitaskingSupported`

  property of the **`UIDevice`** class to determine whether multitasking is available before performing the relevant task.

```
let device = UIDevice.current
if device.isMultitaskingSupported
{
    // ... Any additional code ...
}
```

App States and Multitasking

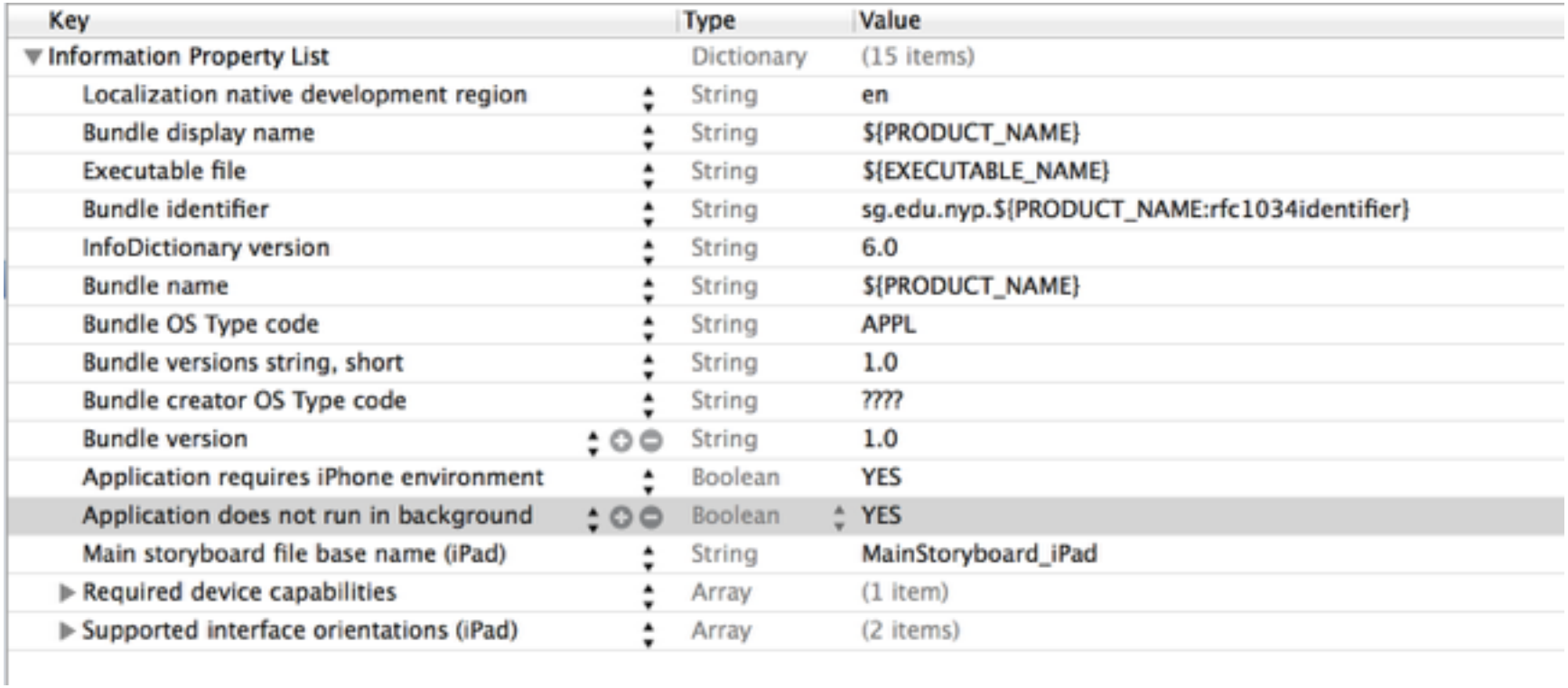
Background Execution

Background Execution

- When an iOS application goes into Background:
  - Max **3 minutes** to finish up tasks before suspension
    (if you specifically request this from iOS)
  - After that, your app will not execute anymore code.

Background Execution

- Opt out of Background Execution in Info.plist:



Key	Type	Value
▼ Information Property List	Dictionary	(15 items)
Localization native development region	String	en
Bundle display name	String	${PRODUCT_NAME}
Executable file	String	${EXECUTABLE_NAME}
Bundle identifier	String	sg.edu.nyp.${PRODUCT_NAME:rfc1034identifier}
InfoDictionary version	String	6.0
Bundle name	String	${PRODUCT_NAME}
Bundle OS Type code	String	APPL
Bundle versions string, short	String	1.0
Bundle creator OS Type code	String	????
Bundle version	String	1.0
Application requires iPhone environment	Boolean	YES
Application does not run in background	Boolean	YES
Main storyboard file base name (iPad)	String	MainStoryboard_iPad
▶ Required device capabilities	Array	(1 item)
▶ Supported interface orientations (iPad)	Array	(2 items)

Background App Refresh

- You can declare you want

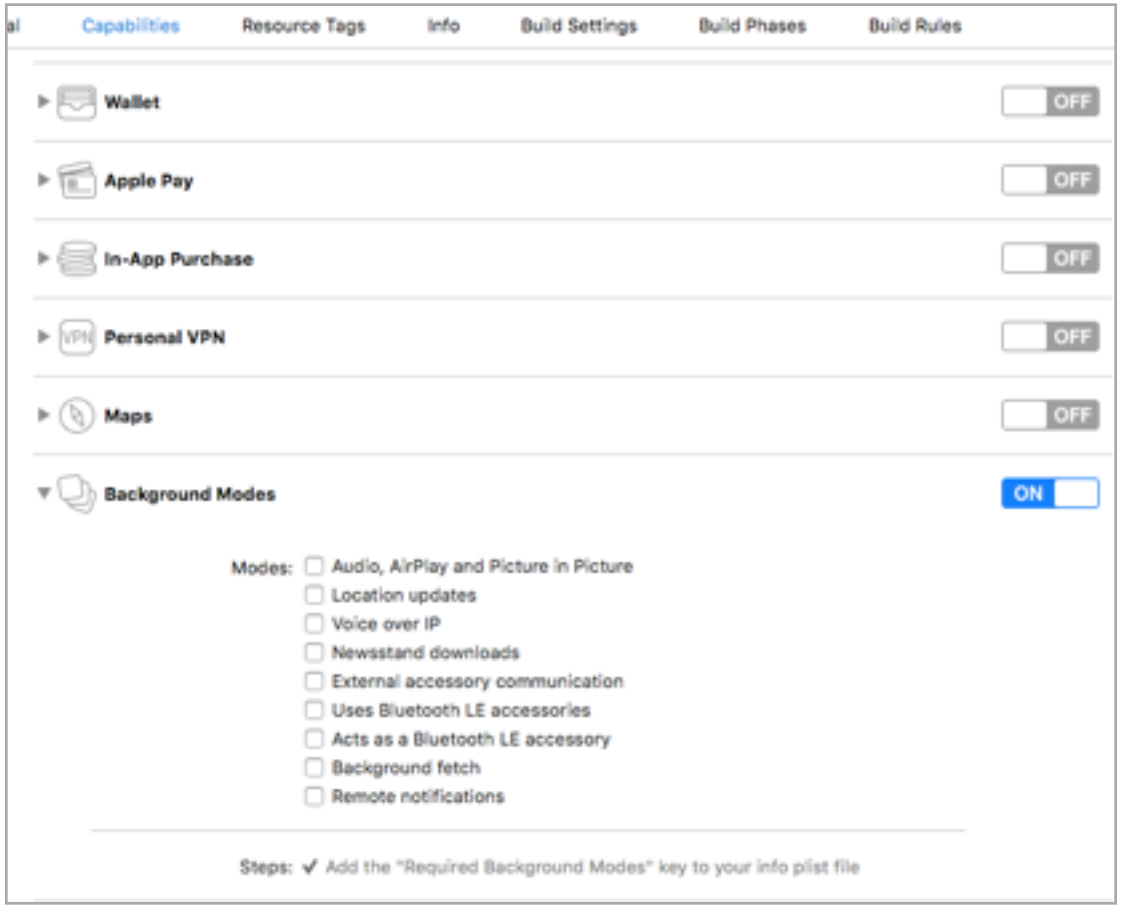
  Background App Refresh

  Means your app does one of:

  - Audio + Airplay
  - Location Updates
  - Voice over IP
  - Newsstand downloads
  - External accessory communication
  - Bluetooth LE
  - Background URL Fetch
  - Remote / Push Notifications (with content download)

Background App Refresh

- Choose Background App Refresh modes for your app:



Background App Refresh

- iOS determines **when** your app will execute a background app refresh. Your app will be launched into Background.

- Your app is given **30 seconds** to run during a background refresh.

- **NOTE:**
  - Only works if the app is in suspended state.
  - Does **not** work if the app is terminated by the user.

App States and Multitasking

Saving / Restoring Application State

Archiving/Unarchiving

- One of the most common ways of persisting objects on iOS.
- Archive an object:
  - save instance variables to filesystem
- Unarchive an object:
  - loads instance variables from filesystem

Archiving/Unarchiving

- Objects can be made archive-able / unarchive-able by implementing **NSCoding** protocol:

```
class Movie: NSObject, NSCoding {

    required init?(coder aDecoder: NSCoder) { }

    func encode(with aCoder: NSCoder) { }
}
```

NSCoder

- NSCoder object as a container for data is responsible for organizing data and writing them to the file system.
- It organizes the data in key-value pairs.

Make Movie conform to the NSCoding

```
class Movie: NSObject, NSCoding
```

Implement encodeWithCoder

```
func encode(with aCoder: NSCoder)
{
    aCoder.encode(movieName ?? "", forKey: "movieName")
    aCoder.encode(movieDesc ?? "", forKey: "movieDesc")
    aCoder.encode(runtime, forKey: "runtime")
    aCoder.encode(imageName ?? "", forKey: "imageName")
}
```

NSCoder

Objects being loaded from an archive:

```
    `

    required init?(coder aDecoder: NSCoder)
    {
        movieName = aDecoder
            .decodeObject(forKey: "movieName") as? String
        movieDesc = aDecoder
            .decodeObject(forKey: "movieDesc") as? String
        runtime = aDecoder.decodeInteger(forKey: "runtime")
        imageName = aDecoder
            .decodeObject(forKey: "imageName") as? String

        super.init()
    }
```

Application Sandbox

- Every iOS application has its own application sandbox.

Directories	Description
application bundle	It contains all the resources and the executable.
Library/Preferences/	It contains any preferences.
tmp/	This directory is where you write data that you will be using temporarily during the application's runtime.
Documents/	This directory is where you write data that the application generates during runtime. It is backed up when the device is synchronized. For example, in a game application, the saved game file would be stored.
Library/Caches	This directory is where you write data that the application generates during runtime. Unlike documents, it does not get backed up when synchronized.

Constructing a file path

```
func itemArchivePath() -> String
{
    let documentDirectories =
        NSSearchPathForDirectoriesInDomains(
        .documentDirectory, .userDomainMask, true)

    let documentDirectory = documentDirectories[0]

    return (documentDirectory as NSString)
        .appendingPathComponent("items.archive")

}
```

NSSearchPathForDirectoriesInDomains searches the filesystem for a path that meets the criteria given by the arguments.

- First argument is a constant that specifies the directory in the sandbox you want to path to.
- The last two argument is always the same on iOS.

NSKeyedArchiver and NSKeyedUnarchiver

```
let path = self.itemArchivePath()
let success = NSKeyedArchiver
    .archiveRootObject(movieList, toFile: path)
if (success)
{
    print("Saved all of the items")
}
else
{
    print("Could not save any of the items")
}
```

archiveRootObject(object, toFile) method will archive every item in movieList to the itemArchivePath.

The movieLists array will send encodeWithCoder to all of the objects it contains, passing the same NSKeyArchiver.

NSKeyedArchiver and NSKeyedUnarchiver

```
let movieListUnarchived = NSKeyedUnarchiver
    .unarchiveObjectWithFile(itemArchivePath())
```

To load the movieList when the application launches, we will use NSKeyUnarchiver.

The unarchiveObjectWithFile: method will create an instance of NSKeyUnarchiver and load the archive located at the itemArchivePath.

Summary

- App States and Multitasking
- Application Lifetime Events
- Background Execution
- Saving and Restoring Application States

Reference: https://developer.apple.com/library/ios/documentation/iphone/conceptual/iPhoneOSProgrammingGuide/TheAppLifeCycle/TheAppLifeCycle.html#//apple_ref/doc/uid/TP40007072-CH2-SW1