

---

# UserDefault

---

# 데이터 저장 방법

---

- 파일 저장 (**Property List**, SQLite, 아카이빙)
- iOS DataBase이용 - Core Data
- Network - Server DB이용

# UserDefaults

---

- 사용자의 정보를 저장하는 싱글톤 인스턴스(싱글톤에 대해선 나중에 자세히)
- 간단한 사용자 정보를 저장/불러오기가 가능하게 만든 인스턴스이다.
- 내부적으로 Plist 파일로 저장되어 보안이 강하지 않다.(필요에 따라 암호화 필요)

# UserDefaults

---

- 주요 항목

```
open class var standard: UserDefaults { get }
```

```
//데이터 불러오기
```

```
open func object(forKey defaultName: String) -> Any?
```

```
open func string(forKey defaultName: String) -> String?
```

```
open func array(forKey defaultName: String) -> [Any]?
```

```
//데이터 저장하기
```

```
open func set(_ value: Any?, forKey defaultName: String)
```

```
//데이터 지우기
```

```
open func removeObject(forKey defaultName: String)
```

# UserDefaults 예제

---

//\*key값은 통일 시켜야 한다.

//데이터 저장

```
UserDefaults.standard.set("joo", forKey: "userID")
```

//데이터 불러오기

```
let aUser:String = UserDefaults.standard.object(forKey: "UserID") as! String  
let sUser:String = UserDefaults.standard.string(forKey: "UserID")!
```

# 실습 : UserDefaults

---

- UserDefaults를 사용해서 이름, 나이 데이터 저장하기
- 레이블에 정보 표시하기

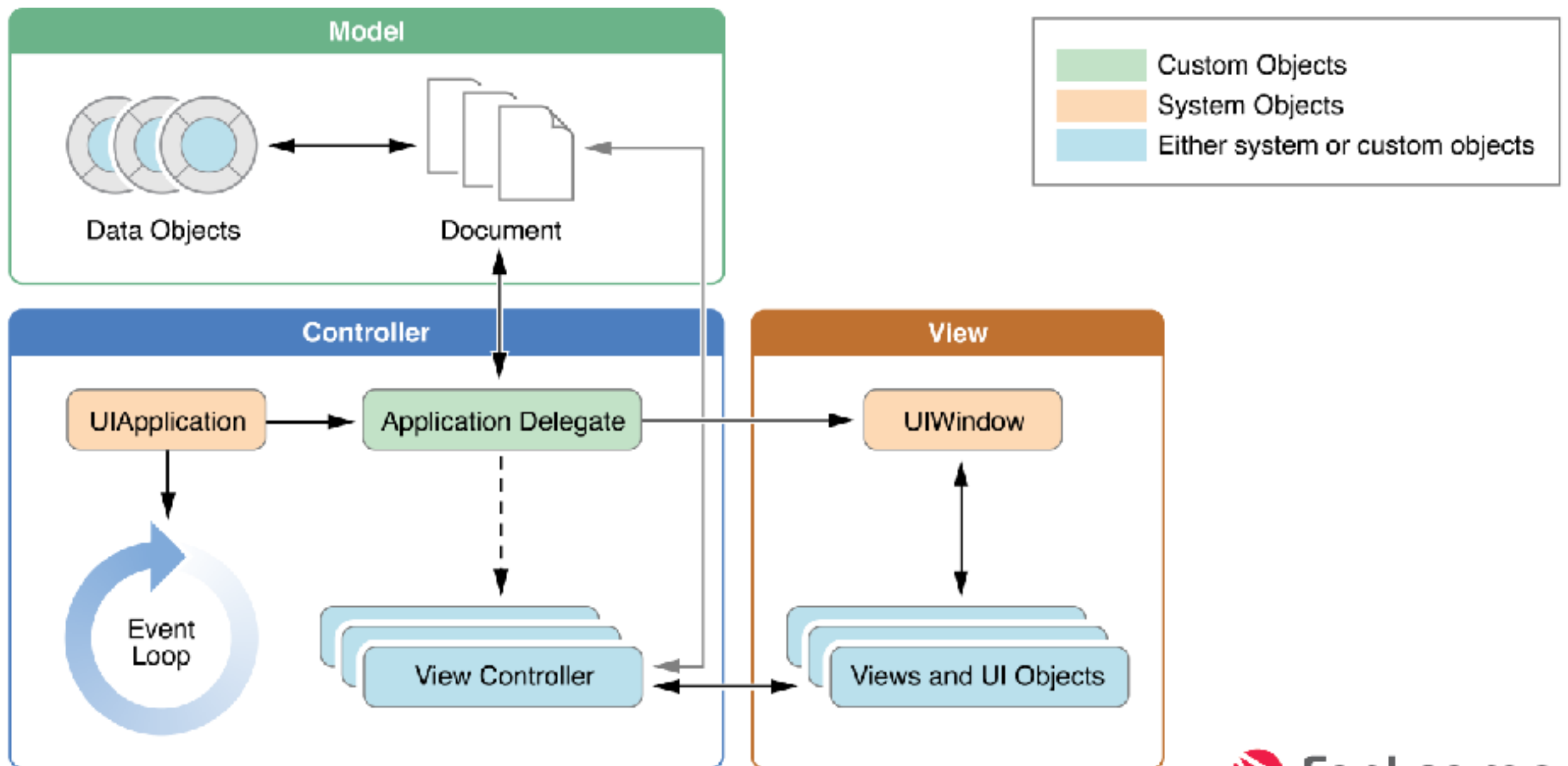
---

# Application Life Cycle

---

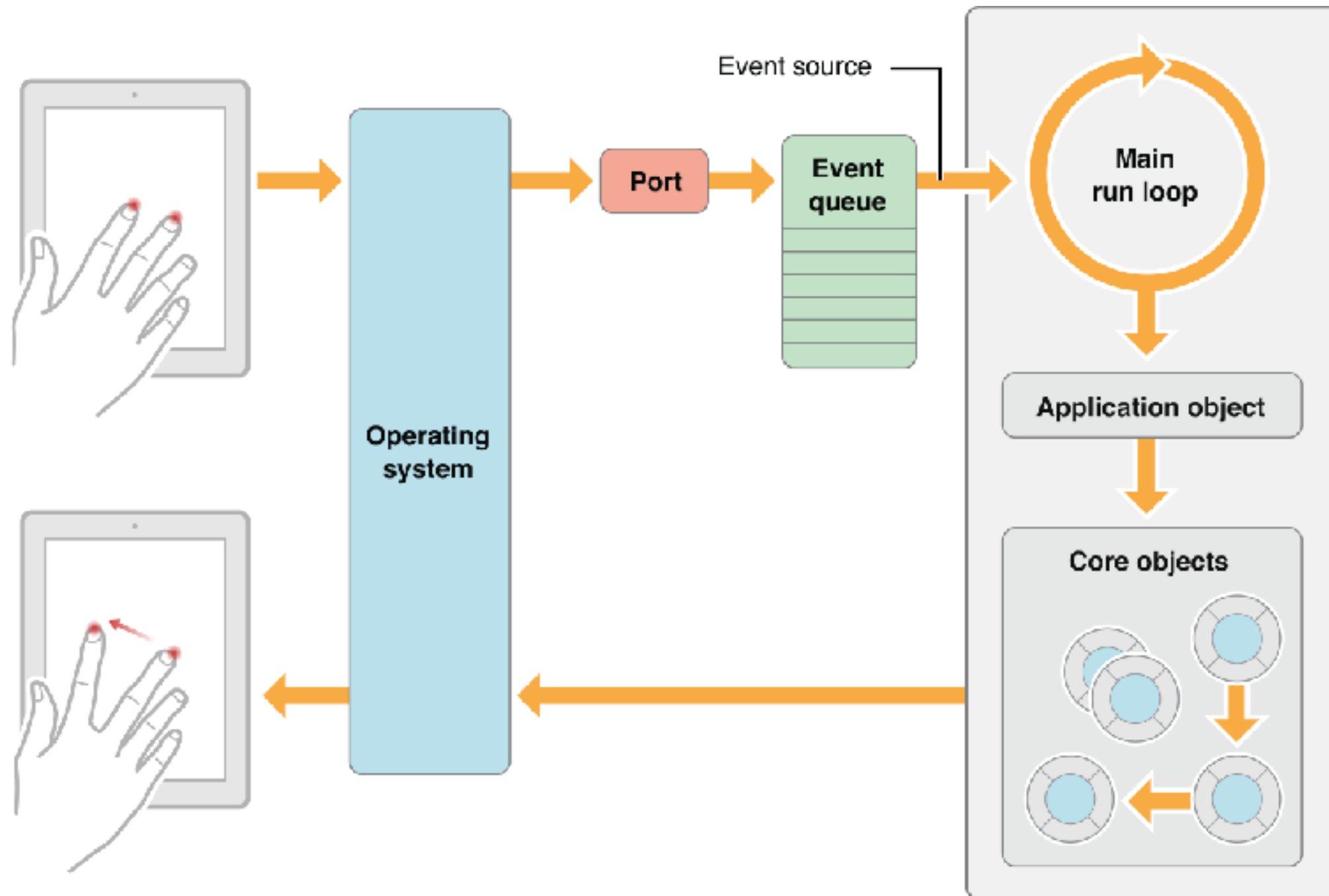
# The Structure of an App

During startup, the UIApplicationMain function sets up several key objects and starts the app running. At the heart of every iOS app is the UIApplication object





# The Main Run Loop



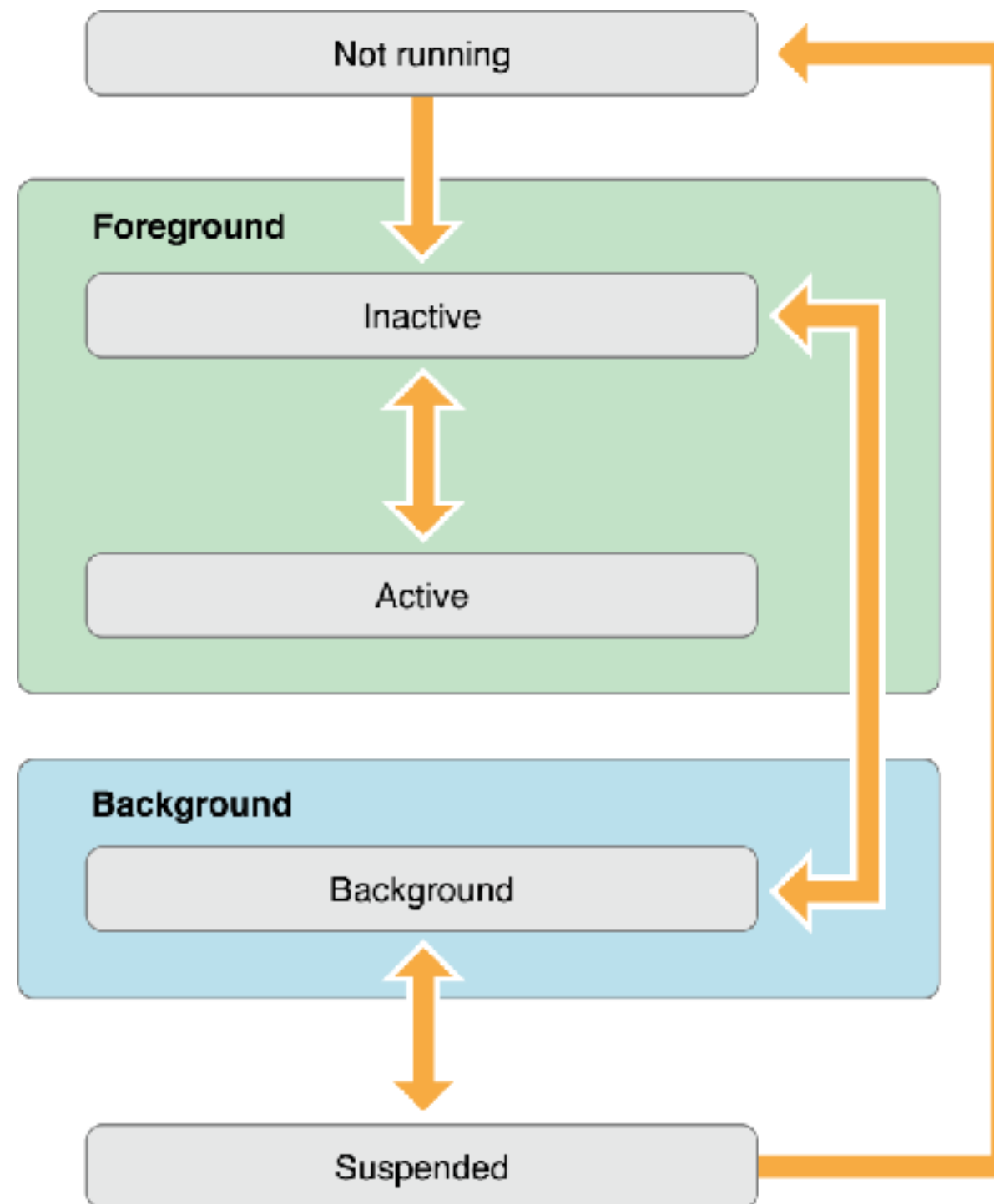
# Event에 대한 처리

---

- Touch : 발생한 이벤트에 대한 뷰가 처리
- Remote control & Shake motion events : First responder객체
- Accelerometer/Magnetometer/Gyroscope : 각각의 객체로 전달
- Location :CoreLocation 객체
- Redraw : 업데이트를 원하는 뷰가 처리

# Execution States for Apps

---



# Execution States for Apps

- Not Running : 실행되지 않았거나, 시스템에 의해 종료된 상태
- Inactive : 실행 중이지만 이벤트를 받고있지 않은 상태. 예를들어, 앱 실행 중 미리알림 또는 일정 알럿이 화면에 덮여서 앱이 실질적으로 이벤트를 받지 못하는 상태 등을 뜻합니다.
- Active : 어플리케이션이 실질적으로 활동하고 있는 상태.
- Background : 백그라운드 상태에서 실질적인 동작을 하고 있는 상태. 예를 들어 백그라운드에서 음악을 실행 하거나, 걸어온 길을 트래킹 하는 등의 동작을 뜻합니다.
- Suspended : 백그라운드 상태에서 활동을 멈춘 상태. 빠른 재실행을 위하여 메모리에 적재된 상태이지만 실질적으로 동작하고 있지는 않습니다. 메모리가 부족할 때 비로소 시스템이 강제종료하게 됩니다.

# Execution States for Apps

---

- Not Running : 실행되지 않았거나, 시스템에 의해 종료된 상태
- Inactive : 실행 중이지만 이벤트를 받고있지 않은 상태. 예를들어, 앱 실행 중 미리알림 또는 일정 알럿이 화면에 덮여서 앱이 실질적으로 이벤트를 받지 못하는 상태 등을 뜻합니다.
- Active : 어플리케이션이 실질적으로 활동하고 있는 상태.
- Background : 백그라운드 상태에서 실질적인 동작을 하고 있는 상태. 예를들어 백그라운드에서 음악을 실행 하거나, 걸어온 길을 트래킹 하는 등의 동작을 뜻합니다.
- Suspended : 백그라운드 상태에서 활동을 멈춘 상태. 빠른 재실행을 위하여 메모리에 적재된 상태이지만 실질적으로 동작하고 있지는 않습니다. 메모리가 부족할 때 비로소 시스템이 강제종료하게 됩니다.

<출력용>

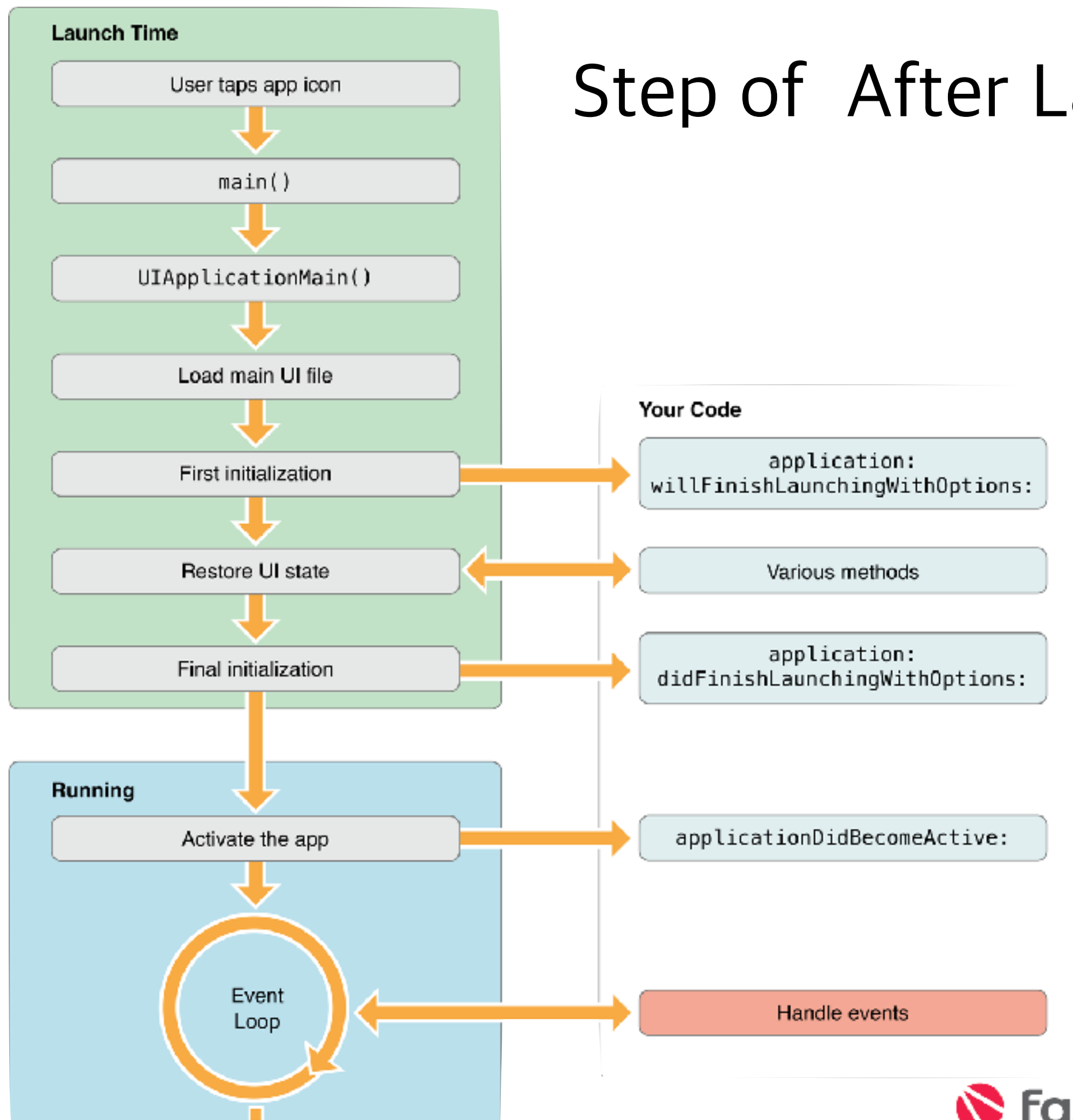
# Call to the methods of your app delegate object

---

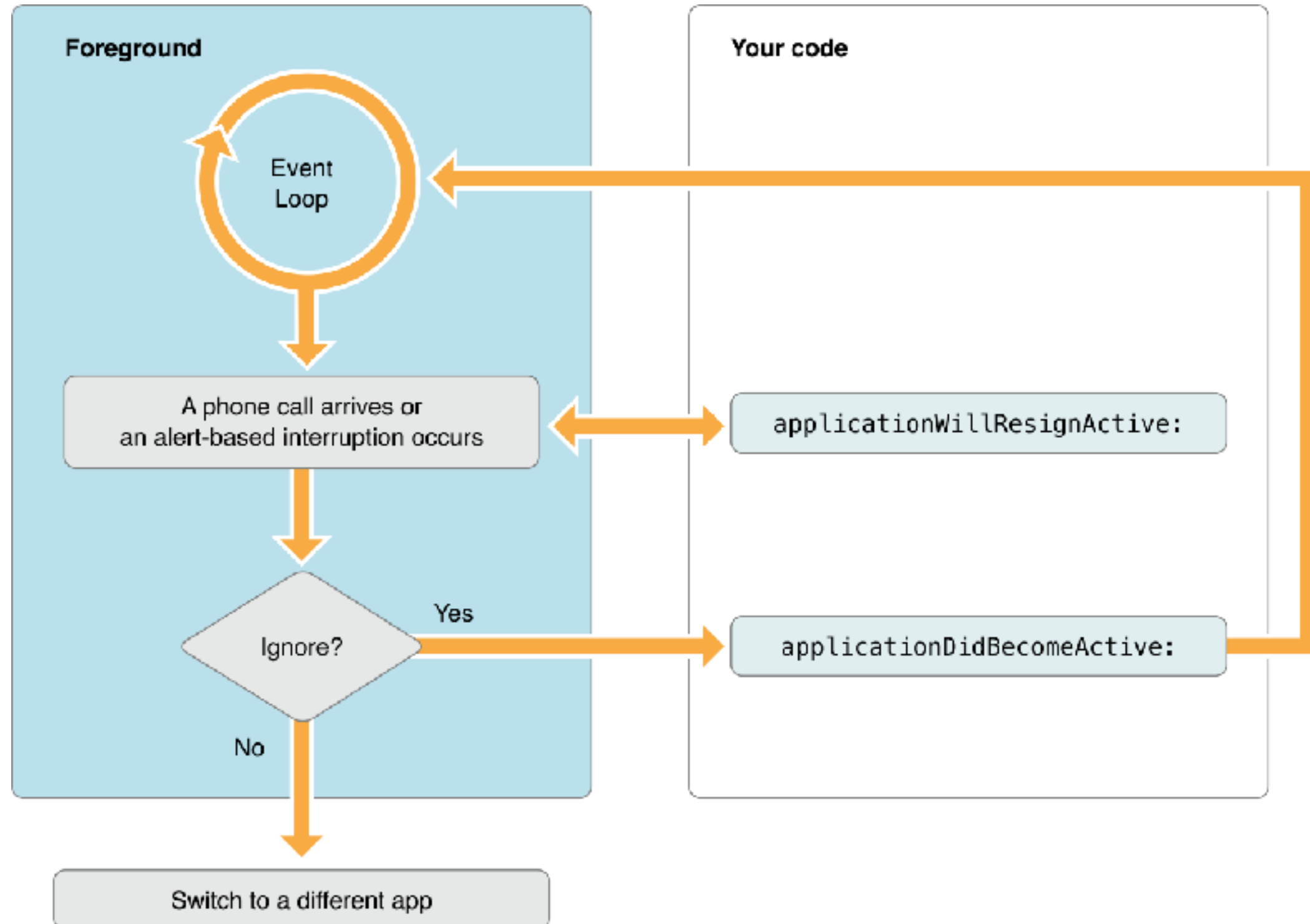
대부분의 상태변화를 app delegate 객체에 호출되는 메소드를 오버라이드하여 알아챌 수 있습니다.

- `application:willFinishLaunchingWithOptions:`
  - 어플리케이션이 최초 실행될 때 호출되는 메소드
- `application:didFinishLaunchingWithOptions:`
  - 어플리케이션이 실행된 직후 사용자의 화면에 보여지기 직전에 호출.
- `applicationDidBecomeActive:`
  - 어플리케이션이 Active 상태로 전환된 직후 호출.
- `applicationWillResignActive:`
  - 어플리케이션이 Inactive 상태로 전환되기 직전 호출
- `applicationDidEnterBackground:`
  - 어플리케이션이 백그라운드 상태로 전환된 직후 호출.
- `applicationWillEnterForeground:`
  - 어플리케이션이 Active 상태가 되기 직전에, 화면에 보여지기 직전의 시점에 호출.
- `applicationWillTerminate:`
  - 어플리케이션이 종료되기 직전에 호출.

# Step of After Launch

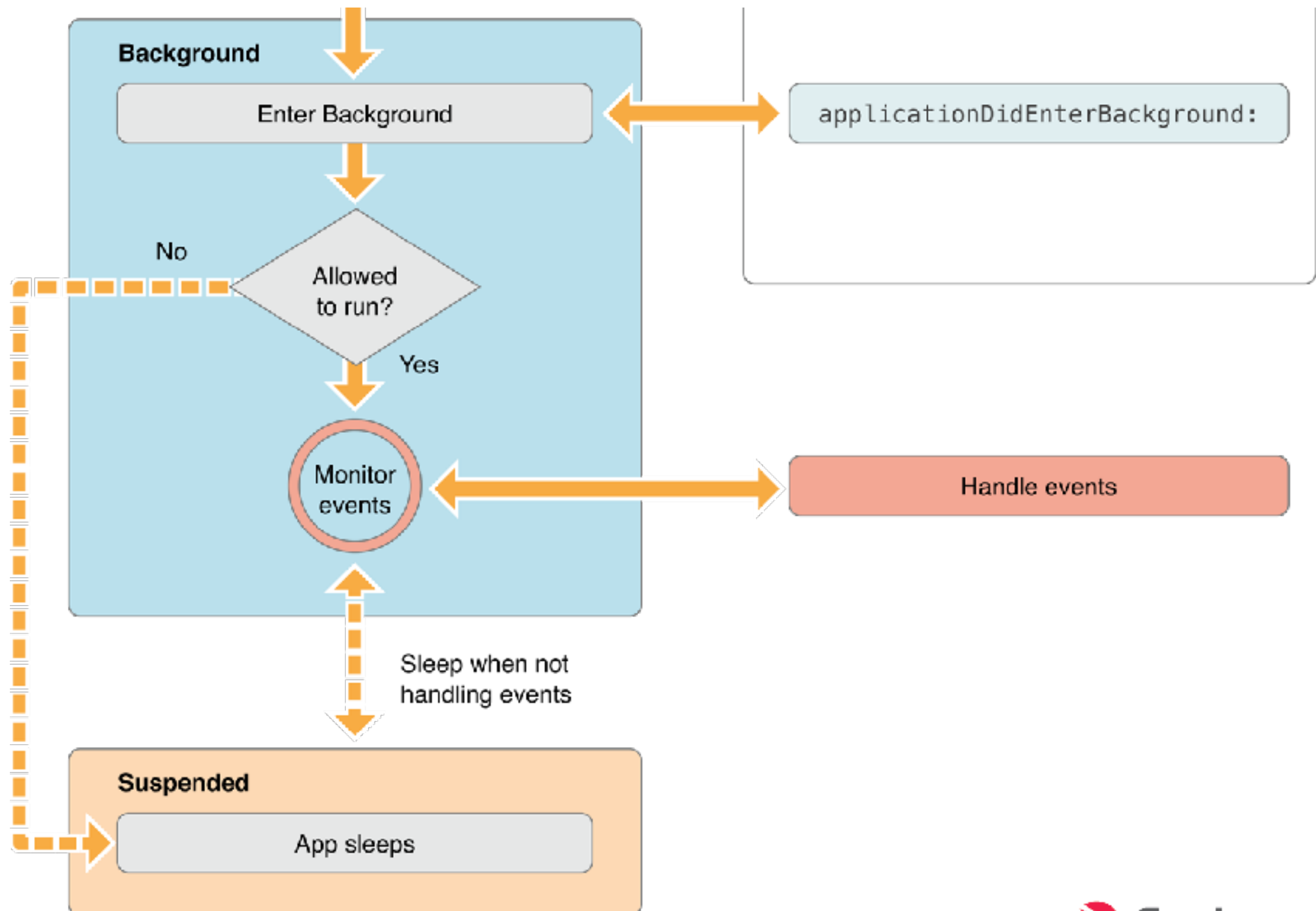


# Step of Interruptions

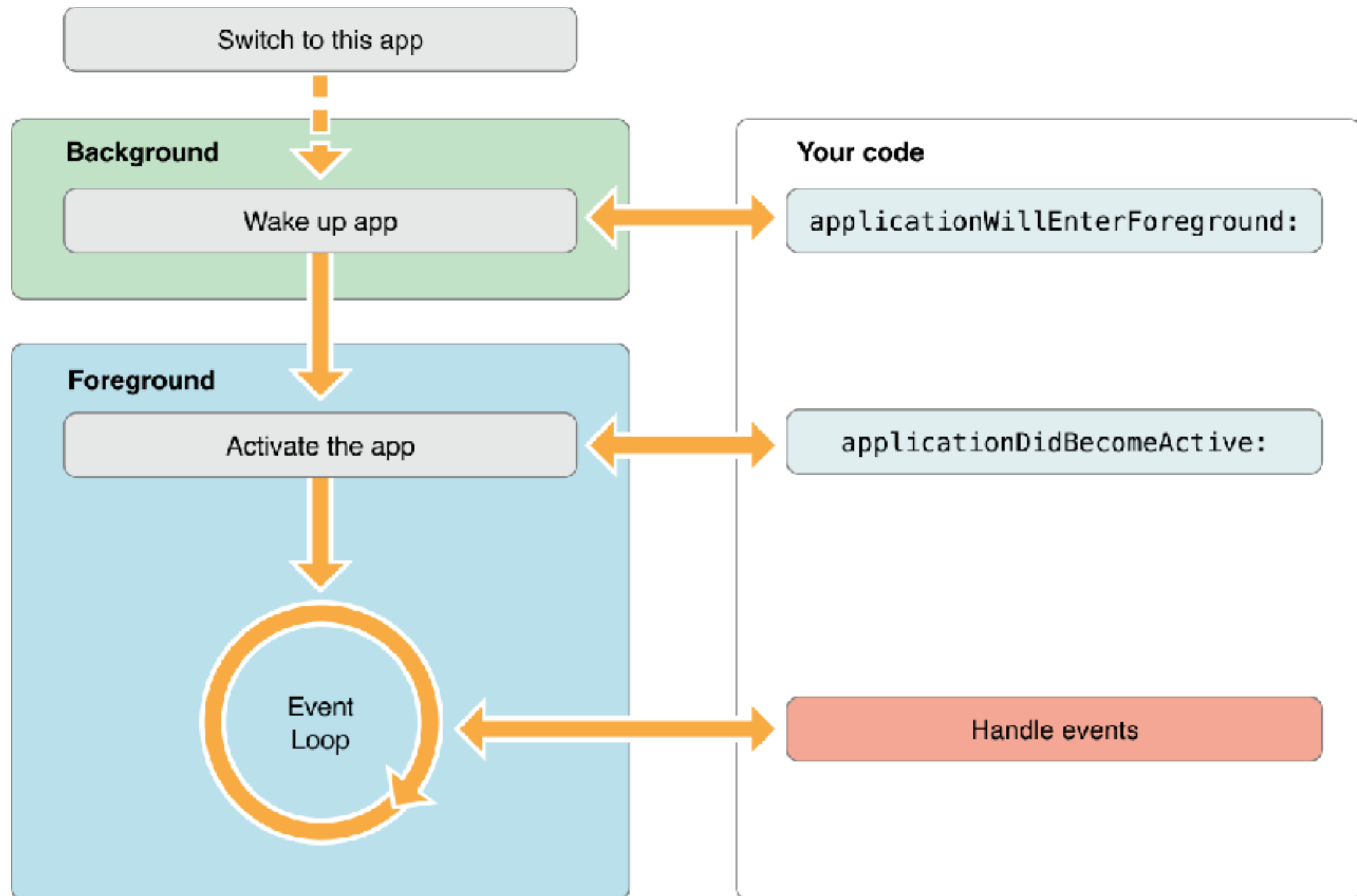




# Step of Enter Background



# Step of Enter Foreground



# Supported Background Tasks

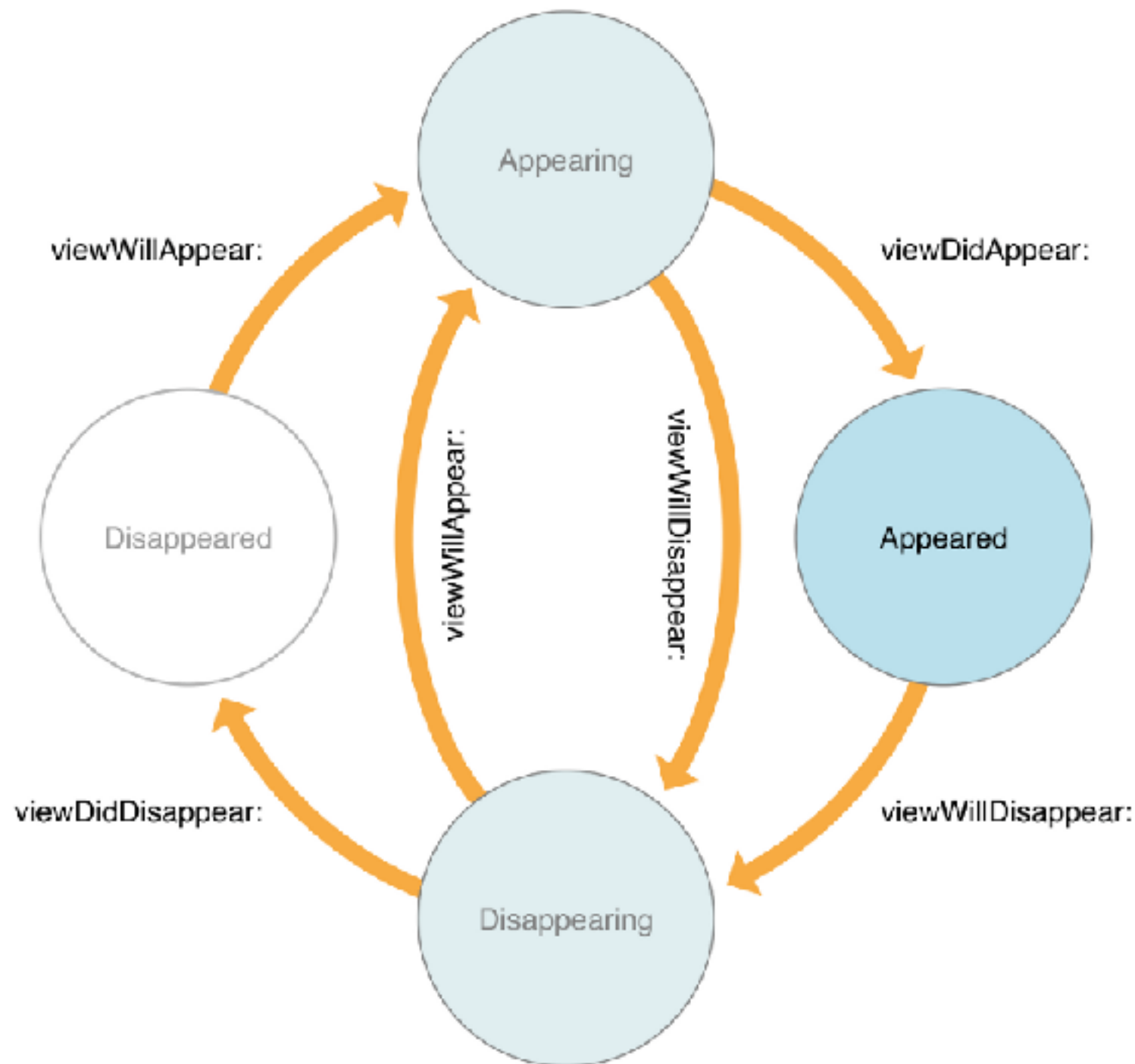
---

- Audio and AirPlay (음악)
- Location updates (위치 정보)
- Voice over IP (인터넷을 사용한 음성통화)
- Newsstand downloads(뉴스 스탠드 다운로드)
- External accessory communication (기타 하드웨어 액세서리)
- Bluetooth LE accessories (블루투스 액세서리 사용)
- Background fetch (네트워크를 통한 일반적인 다운로드나 미완료된 작업)
- Remote notifications (PushNotification)

# UIViewController의 생명주기 메소드

---

- 프로그래머가 직접 호출 불가
- 오버라이드 하는 메소드이므로 꼭 해당 메소드 내에서 [super 메소드]을 통해 기존 메소드를 꼭 호출해야 된다.



# 생명주기 메소드

---

`override func loadView()` : UIViewController의 view가 생성될 때 호출

`override func viewDidLoad()` :

UIViewController가 인스턴스화 된 직후(메모리에 객체가 올라간 직후) 호출 처음 한 번 세팅해 줘야 하는 값들을 넣기에 적절

`override func viewWillAppear(_ animated: Bool)` :

view가 화면에 보여지기 직전에 호출 화면이 보여지기 전에 준비할 때 사용.

animated 파라미터는 뷰가 애니메이션을 동반하여 보여지게 되는지 시스템에서 전달해주는 불리언 값

`override func viewWillLayoutSubviews()` : view의 하위뷰들의 레이아웃이 결정되기 직전 호출

`override func viewDidLayoutSubviews()` :

view의 하위뷰들의 레이아웃이 결정된 후 호출. 주로 view의 하위뷰들이 사이즈 조정이 필요할 때 호출

`override func viewDidAppear(_ animated: Bool)` :

view가 화면에 보여진 직후에 호출. 화면이 표시된 이후 애니메이션 등을 보여주고 싶을 때 유용

`override func viewWillDisappear(_ animated: Bool)` : view가 화면에서 사라지기 직전에 호출

`override func viewDidDisappear(_ animated: Bool)` : view가 화면에서 사라진 직후에 호출

# 확인해 볼까요?

---

- 실제 로그를 찍어 상태를 확인해 봅시다.

---

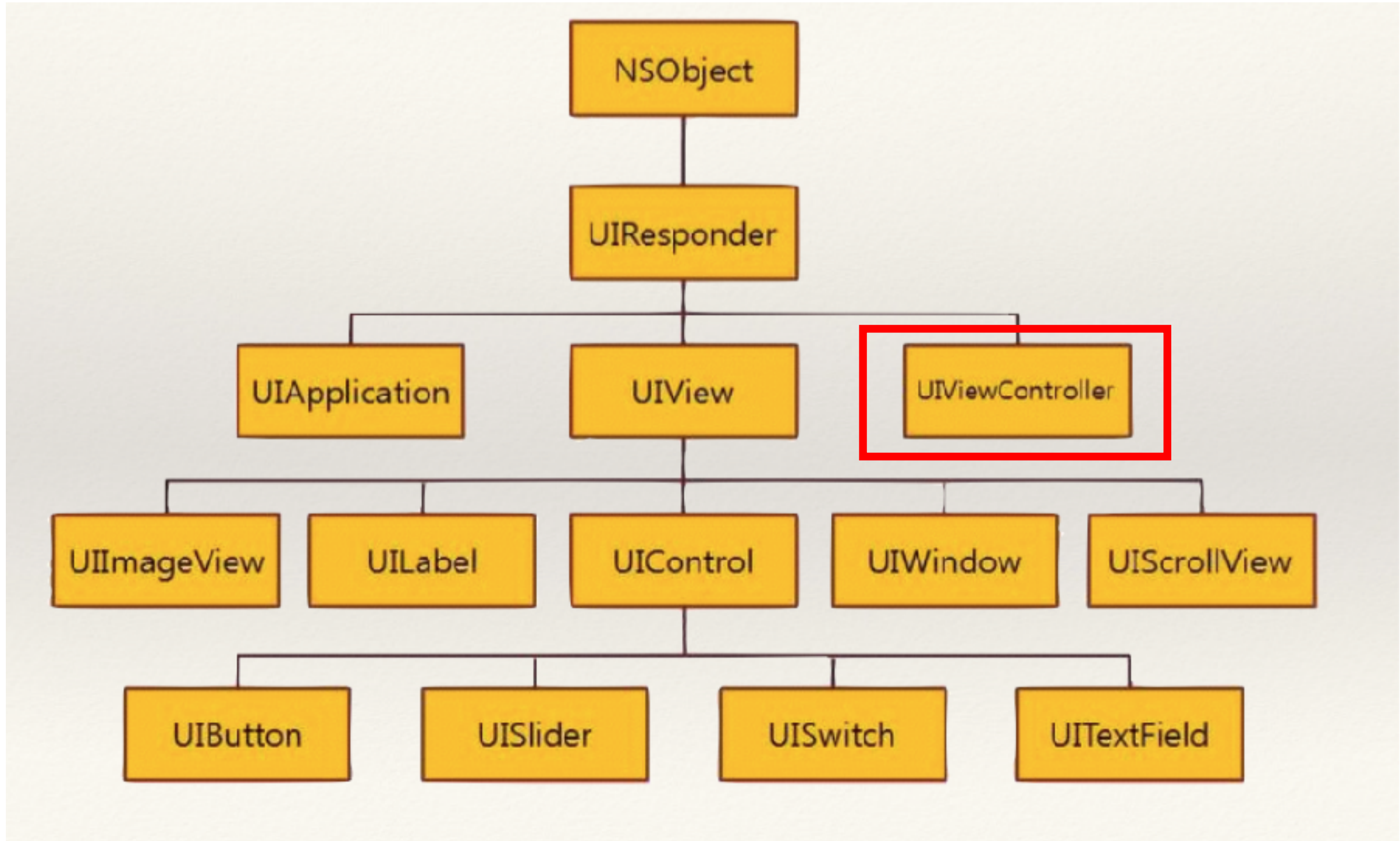
# UIViewController

---

강사 주영민

# UI Class Hierarchy

---





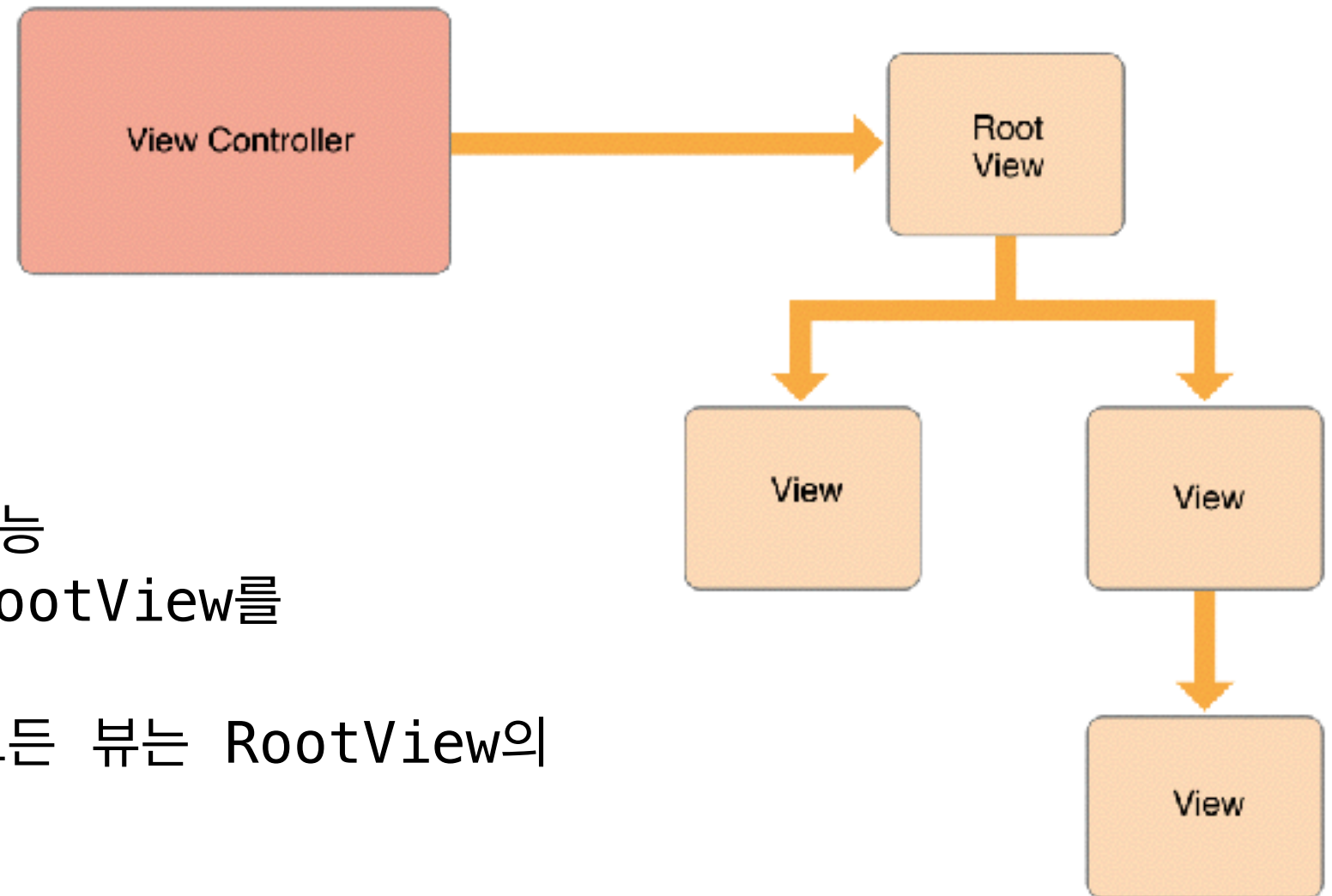
# UIViewController

---

- 앱의 기초가 되는 내부 구조
- 모든 앱은 적어도 한개 이상의 ViewController를 가지고 있으며 대부분의 앱은 여러개의 ViewController로 이뤄져 있다.
- ViewController는 사용자의 인터렉션과 앱의 데이터 사이에서 컨트롤의 역할을 한다.
- ViewController는 View 관리, 사용자 이벤트 핸들링, ViewController간의 전환 등을 위한 메소드와 프로퍼티를 가지고 있다.

# UIViewController - Root View

---



- View의 계층을 관리하는 기능
- 모든 뷰컨트롤러는 한개의 RootView를 가지고 있다.
- 화면에 표시하기 위해서는 모든 뷰는 RootView의 계층 안에 있어야 한다.

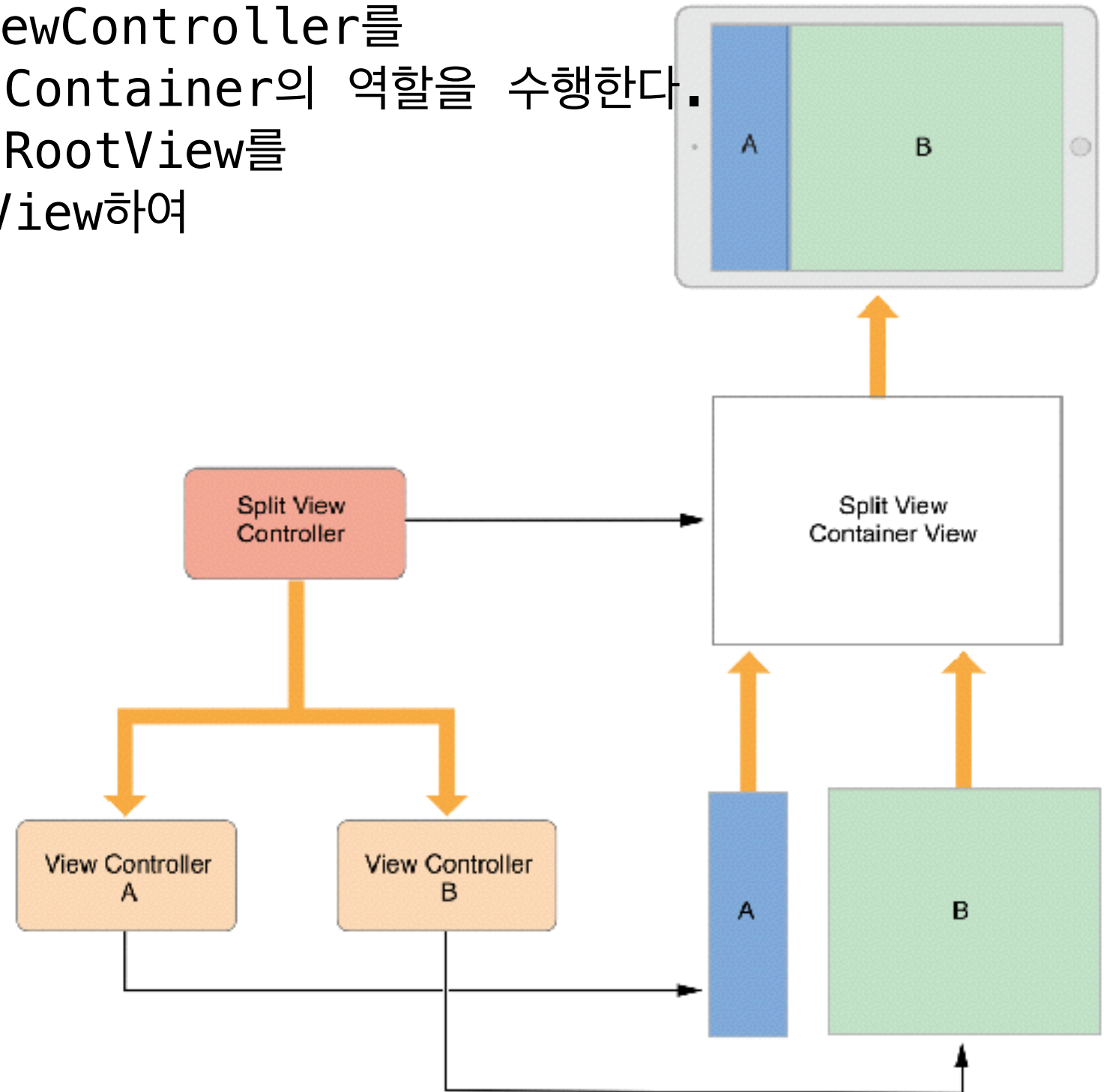
# ViewController 특징

---

- Child UIViewController
- UserInteraction
- Data Marshaling(중계자)
- Resource Management

# Child UIViewController

- ViewController는 다른 ViewController를 Child ViewController로 Container의 역할을 수행한다.
- Child ViewController의 RootView를 자신의 RootView에 addSubview하여 화면에 표시한다.



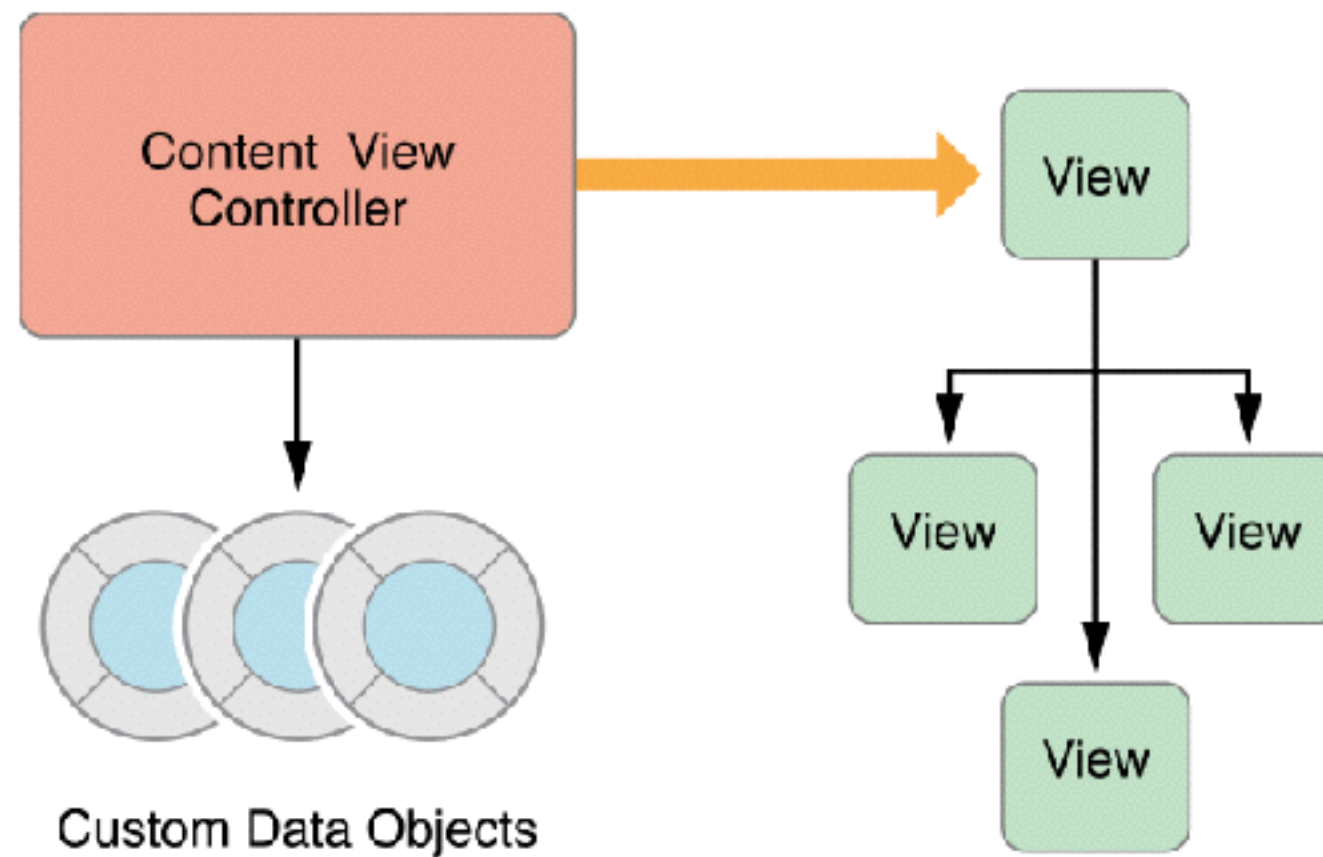
# UserInteraction

---

- UIViewController는 UIResponder를 상속받은 클래스로 이벤트 체인으로부터 오는 이벤트는 효과적으로 처리한다.
- 즉 사용자의 모든 이벤트는 ViewController가 받아서 각 View에 해당되는 Method와 Delegate로 처리한다.

# Data Marshaling(중계자)

- ViewController는 자신이 관리하는 View들과 앱 내부의 데이터와의 중계자 역할을 한다.



# Resource Management

---

- ViewController안에 있는 모든 View나 Object는 모두 ViewController의 책임이다.
- 메모리가 부족시 `didReceiveMemoryWarning()` 메소드가 자동으로 불리며, 오래동안 사용하지 않은 객체와 다시 쉽게 만들수 있는 객체를 제거할수 있어 메모리를 효율적으로 관리한다.

# ViewController 종류

---

- General View Controller
  1. UIViewController
  2. UITableViewController
  3. UICollectionViewController
- Container View Controller
  1. UINavigationController
  2. UITabBarController
  3. UISplitViewController
  4. ....



# General View Controller

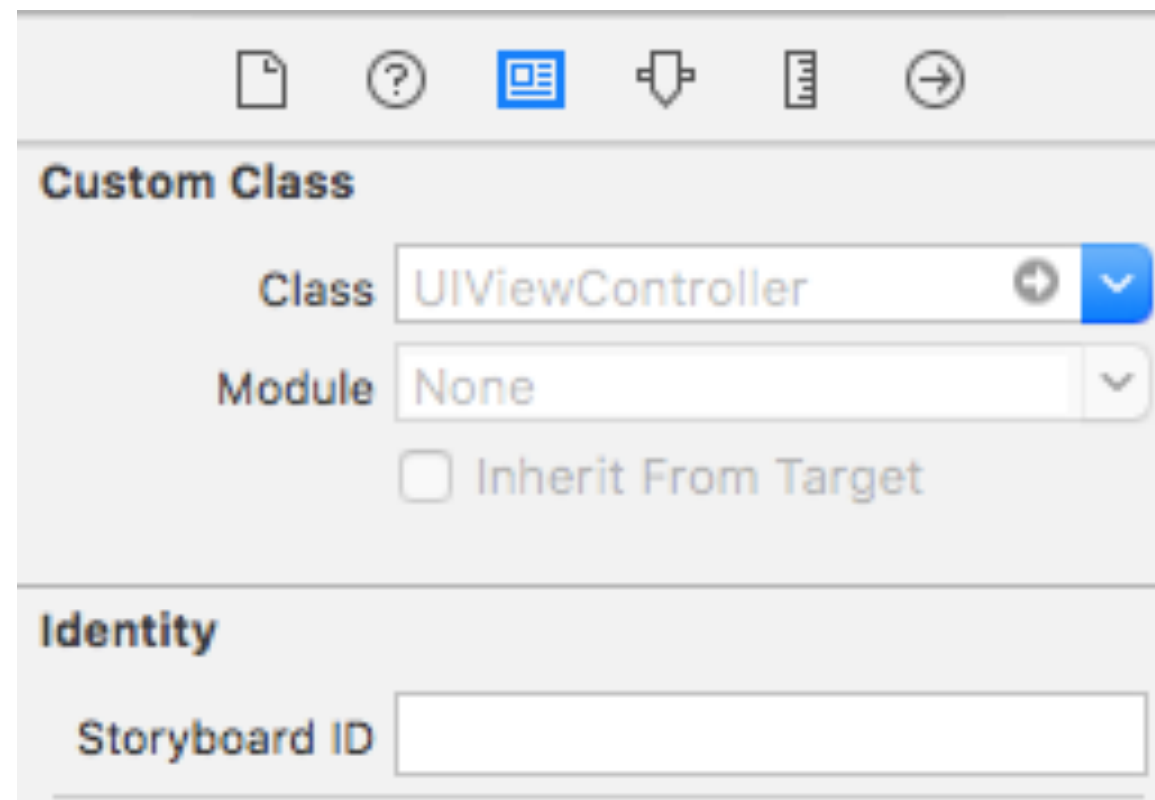
---

- 일반 적인 View Controller 형태
- 각 View Controller가 Root View를 가지고 있다.
  1. UINavigationController Root View = UIView
  2. UITableViewController Root View = UITableView
  3. UICollectionViewController = UICollectionView

# UIViewController Instance Load

---

```
let storyboard = UIStoryboard(name: "Main", bundle: nil)
let vc = storyboard.instantiateViewController(withIdentifier:
"StoryboardID")
```



The screenshot shows the Xcode storyboard editor interface. At the top is a toolbar with icons for file operations, help, storyboard, pin, list, and navigation. Below the toolbar, the 'Custom Class' section is visible, containing a 'Class' dropdown menu set to 'UIViewController', a 'Module' dropdown menu set to 'None', and an unchecked checkbox labeled 'Inherit From Target'. Below this, the 'Identity' section is visible, featuring a 'Storyboard ID' text label and an empty text input field.

# 화면 전환 - Present Modally

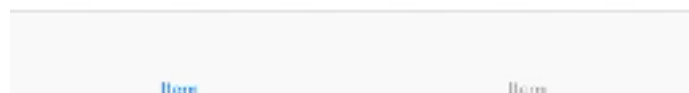
---



Present View Controller

Push View Controller

Present Modally  
ViewController간의 화면 전환



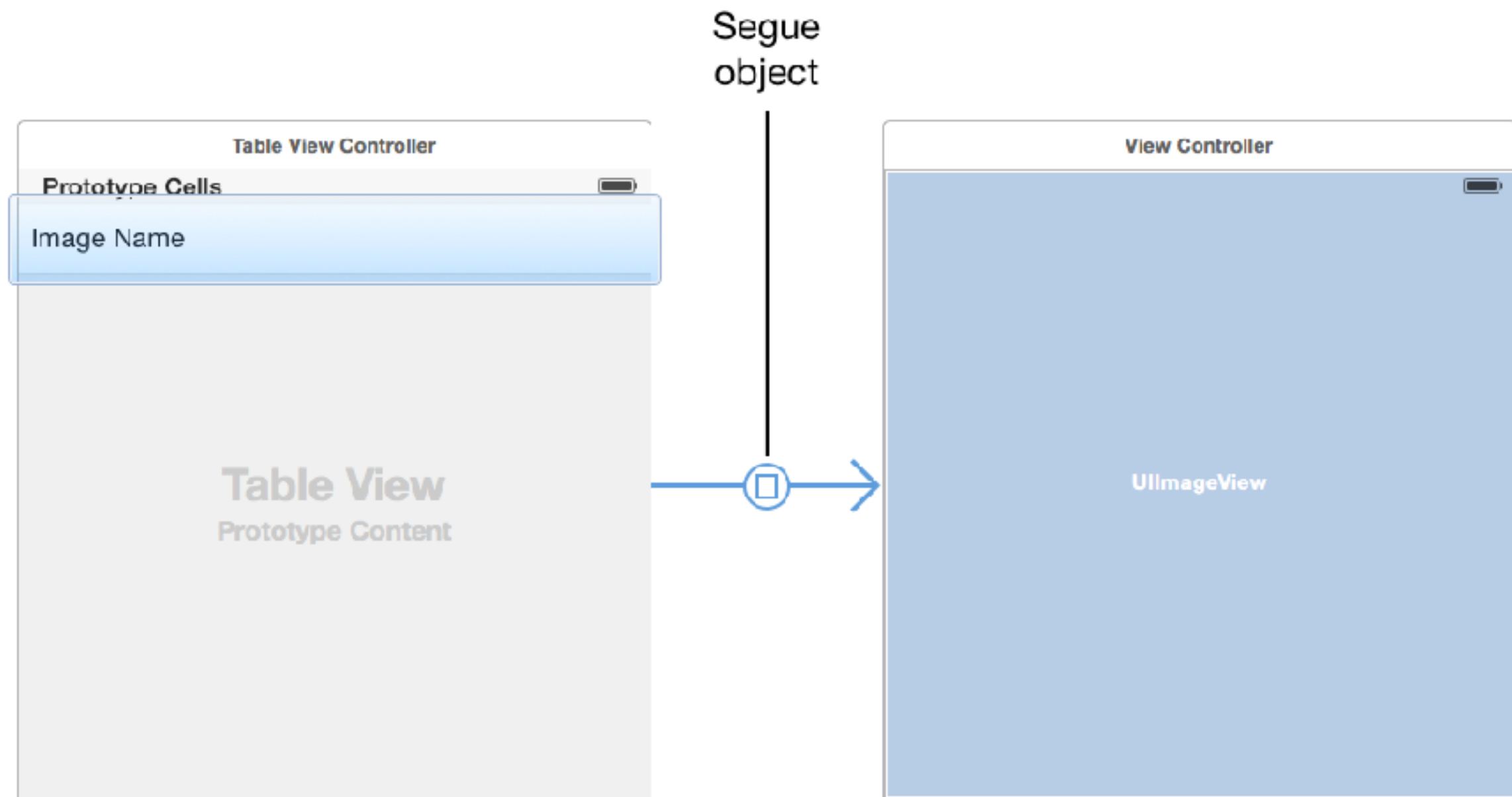
# Segue

---

- 앱의 인터페이스 흐름을 정의하는 데 사용.
- Storyboard 파일 내 두 개의 ViewController사이의 화면전환을 정의.
- Segue의 시작점은 button, tableView의 row, gesture등으로 시작하며 끝점은 전환되는 다음 ViewController이다.
- segue는 일반적으로 다음 ViewController로 진행되는 것을 가르키나, unwind segue를 통해 ViewController를 닫는 역할도 할수 있다.

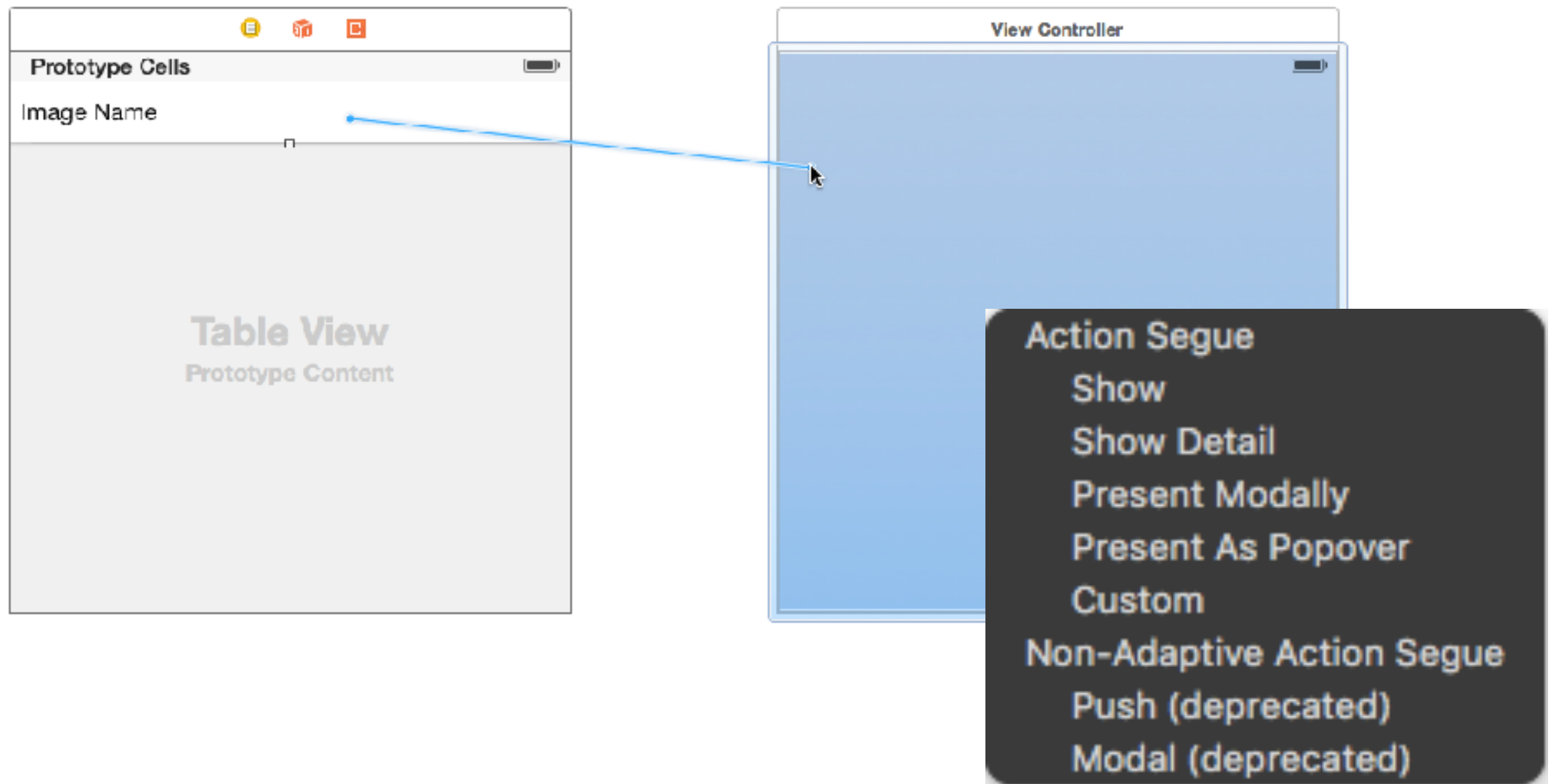
# Segue

---

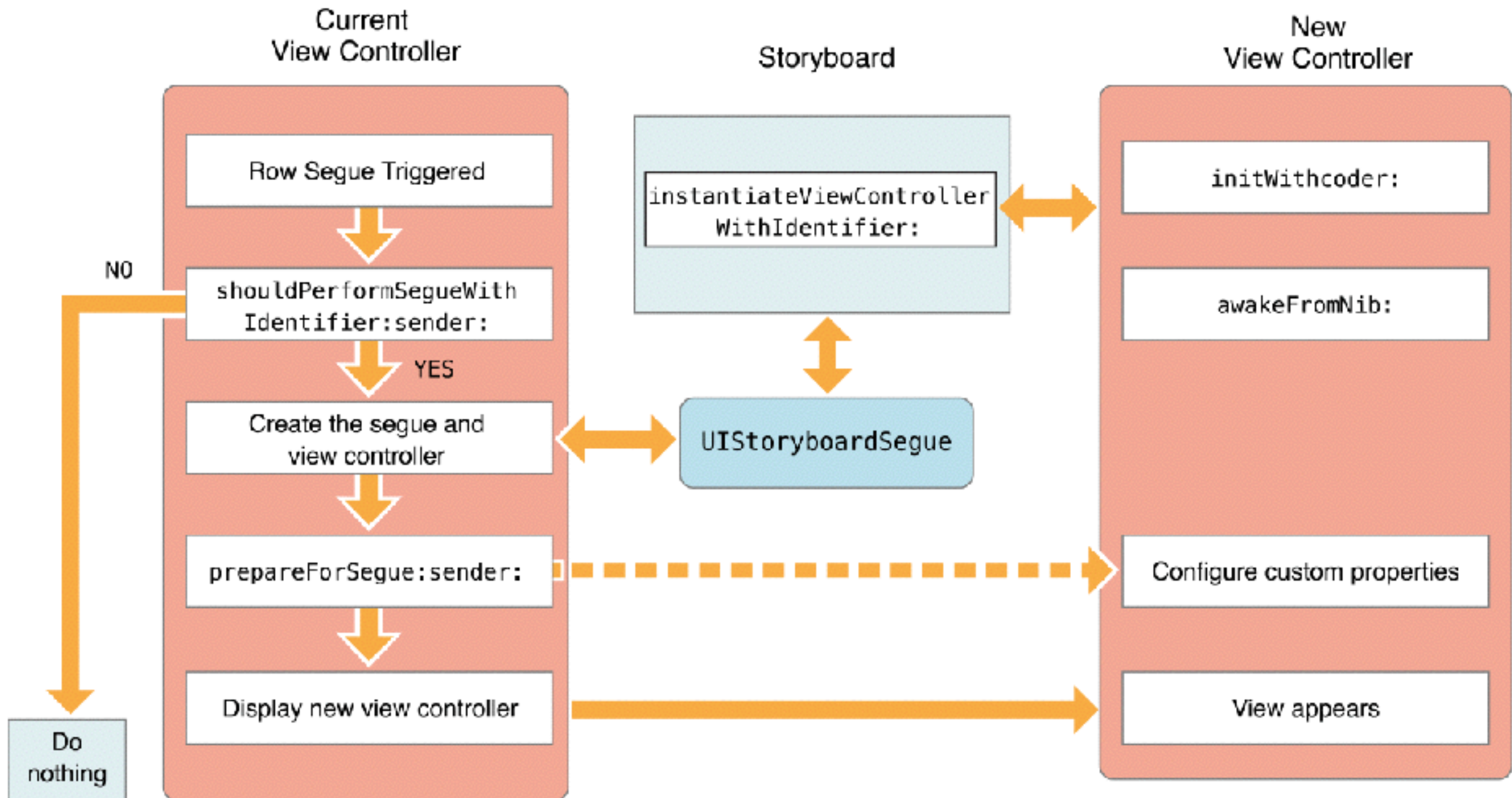


# Create Segue

---



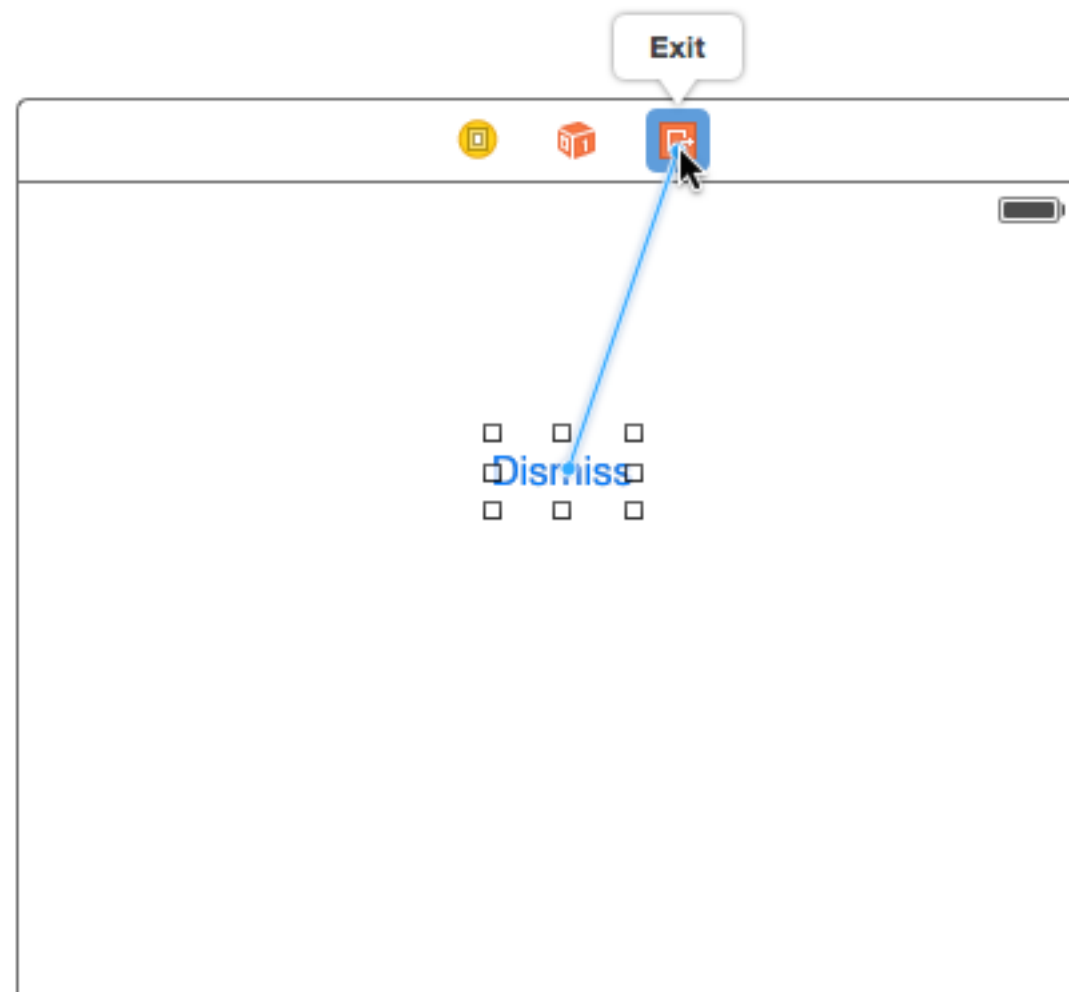
# Modifying a Segue's Behavior at Runtime



# Unwind Segue

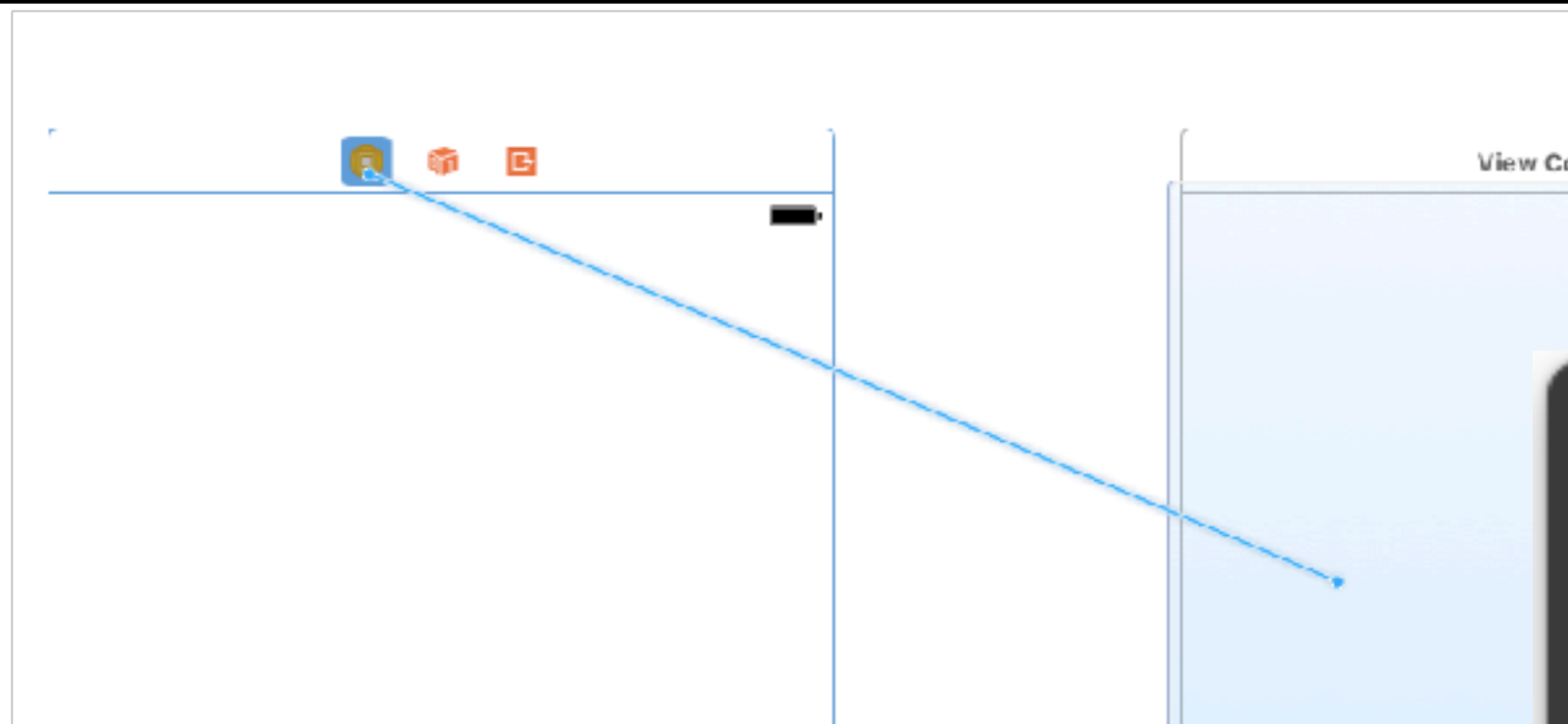
---

1. 되돌아갈 ViewController선택
2. 선택된 ViewController에 다음 메소드 정의
  - (IBAction)myUnwindAction:(UIStoryboardSegue\*)unwindSegue
3. unwind Segue 액션이 필요한 뷰컨트롤러에 액션과 Exit연결





# Manual Segue



Manual Segue

- Show
- Show Detail
- Present Modally
- Present As Popover
- Custom

Non-Adaptive Manual Segue

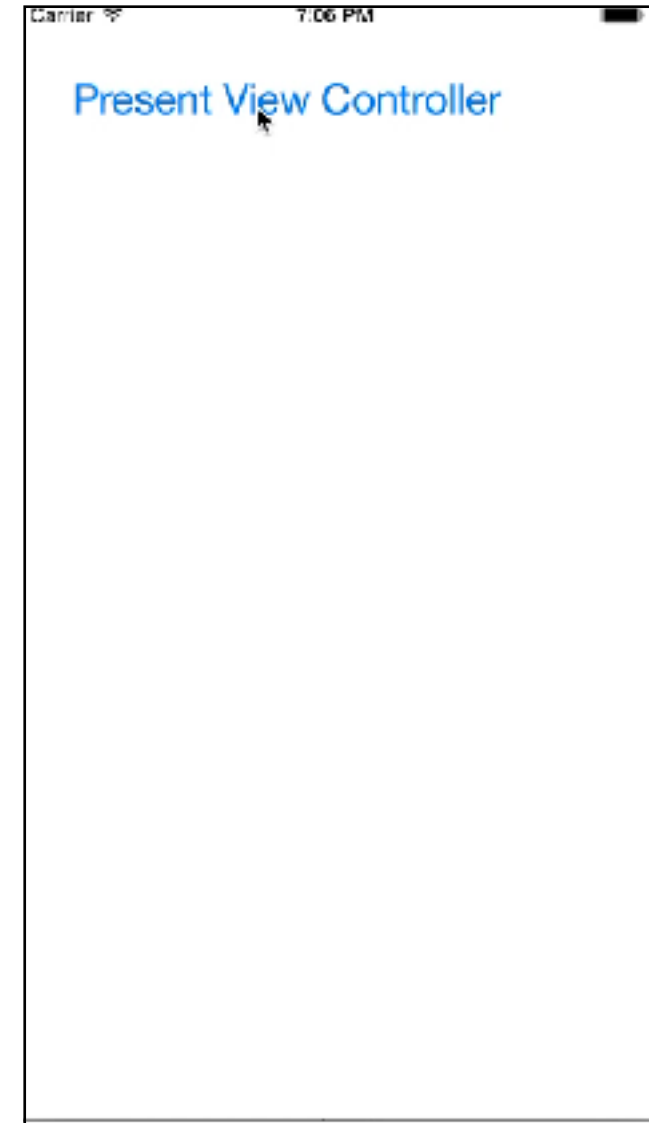
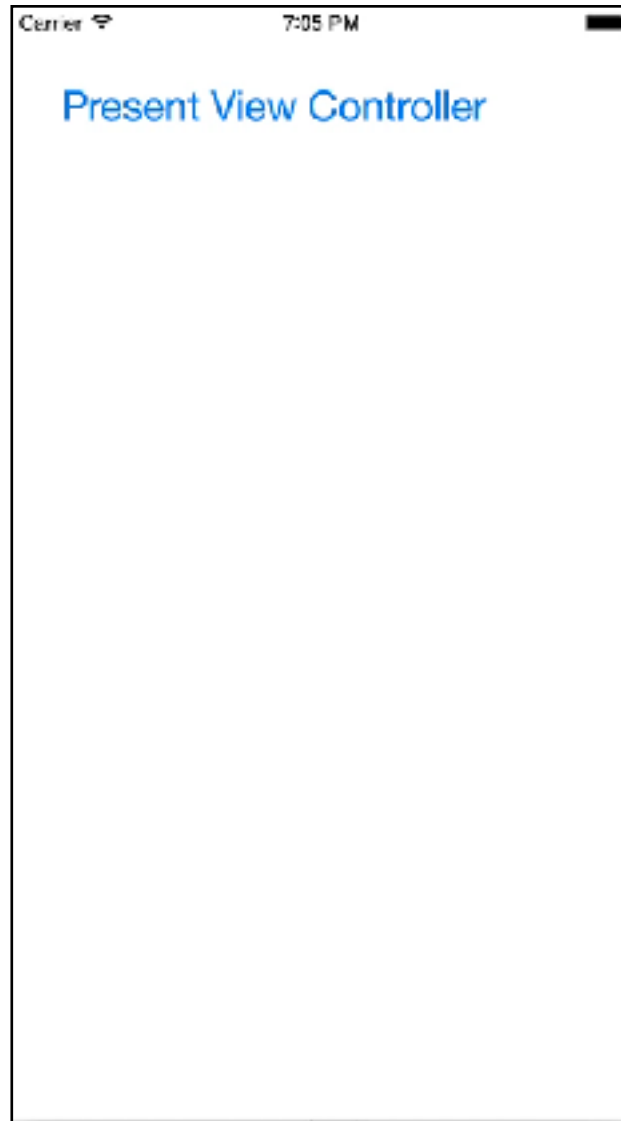
- Push (deprecated)
- Modal (deprecated)

```
[self performSegueWithIdentifier:@"segueName" sender:self];
```

# Animation - transition

---

- animation에 따라 다른 형태의 느낌을 주기도 한다.

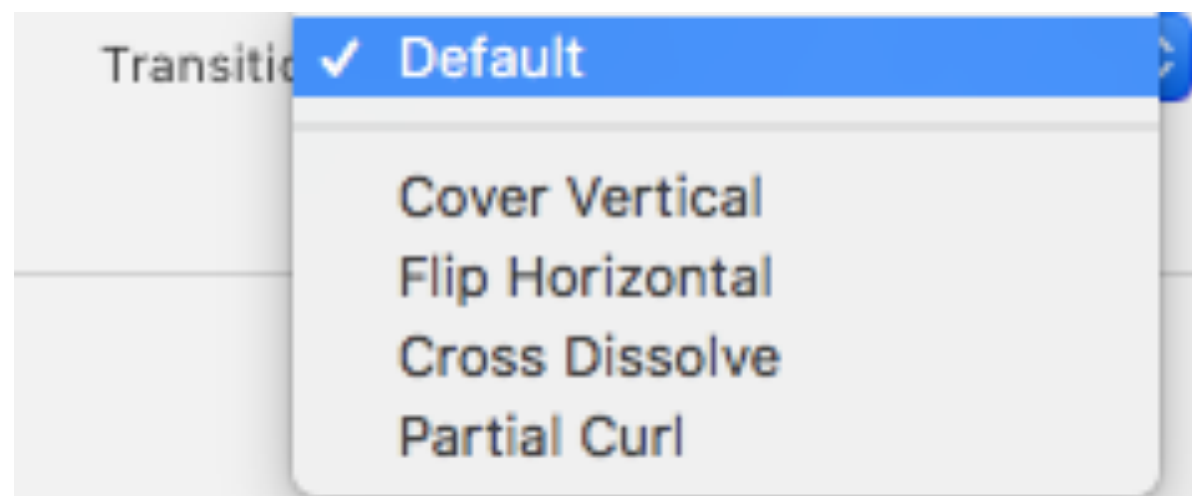


# Animation – transition

---

```
@property(nonatomic,assign) UIModalTransitionStyle modalTransitionStyle;
```

```
typedef NS_ENUM(NSInteger, UIModalTransitionStyle) {  
    UIModalTransitionStyleCoverVertical = 0,  
    UIModalTransitionStyleFlipHorizontal __TVOS_PROHIBITED,  
    UIModalTransitionStyleCrossDissolve,  
    UIModalTransitionStylePartialCurl NS_ENUM_AVAILABLE_IOS(3_2)  
    __TVOS_PROHIBITED,  
};
```



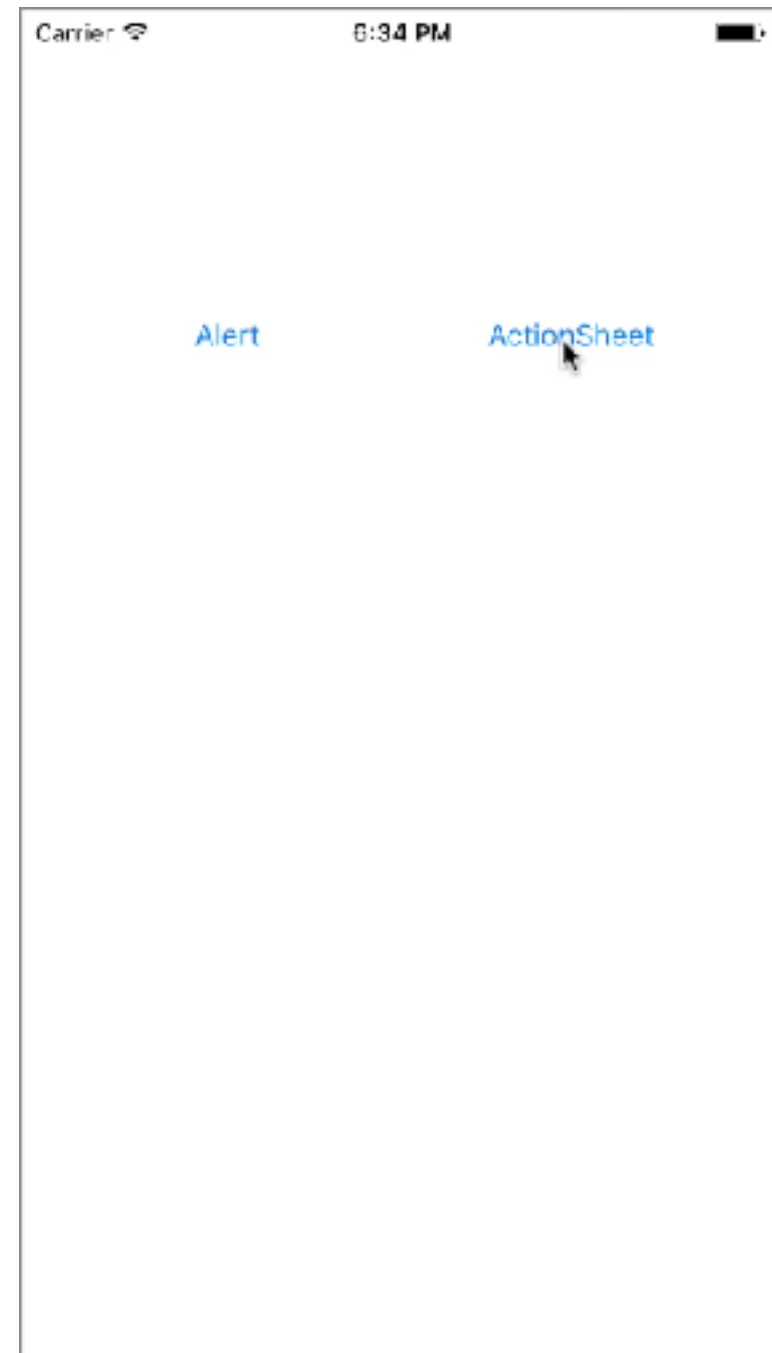
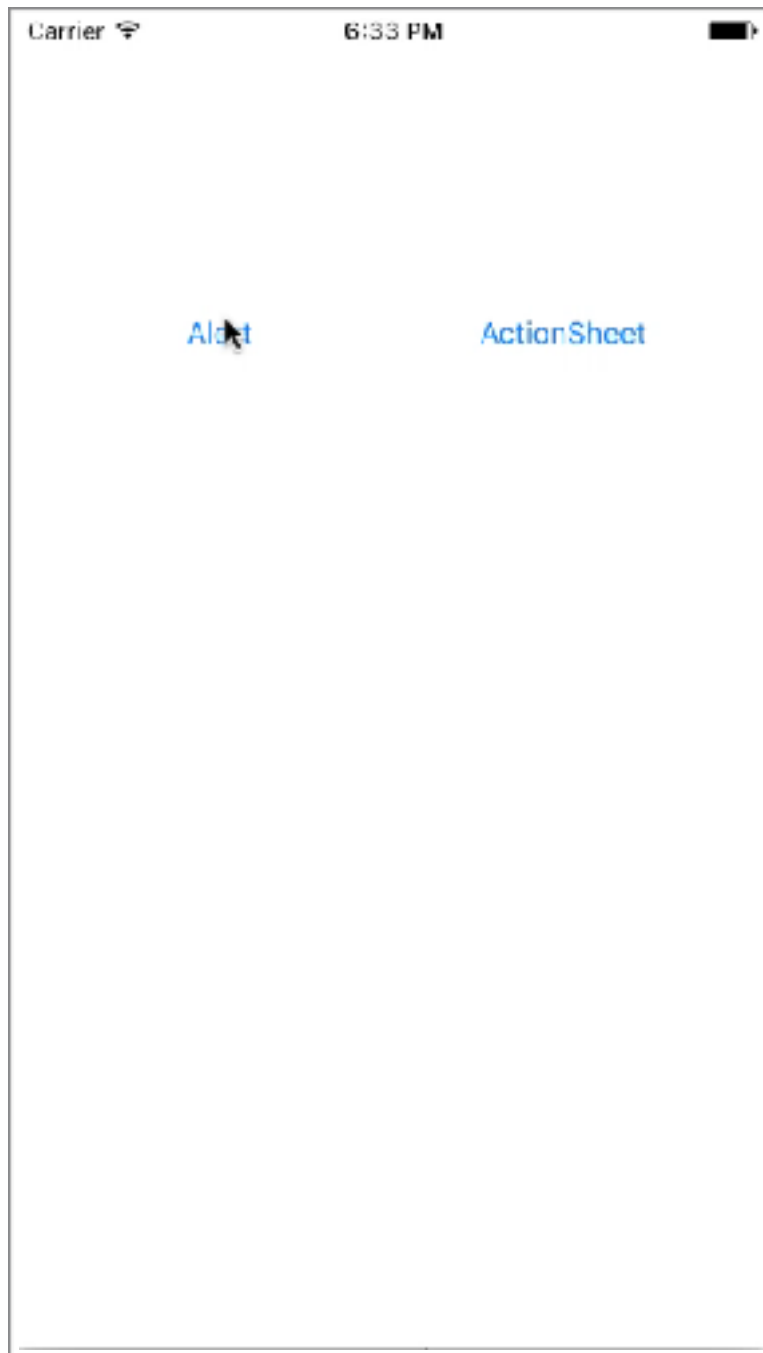
---

# UIAlertController

---

# UIAlertController

---



# Step 1. file 보기

---

- UIAlertController 파일 보기

## Step 2. Sample Code

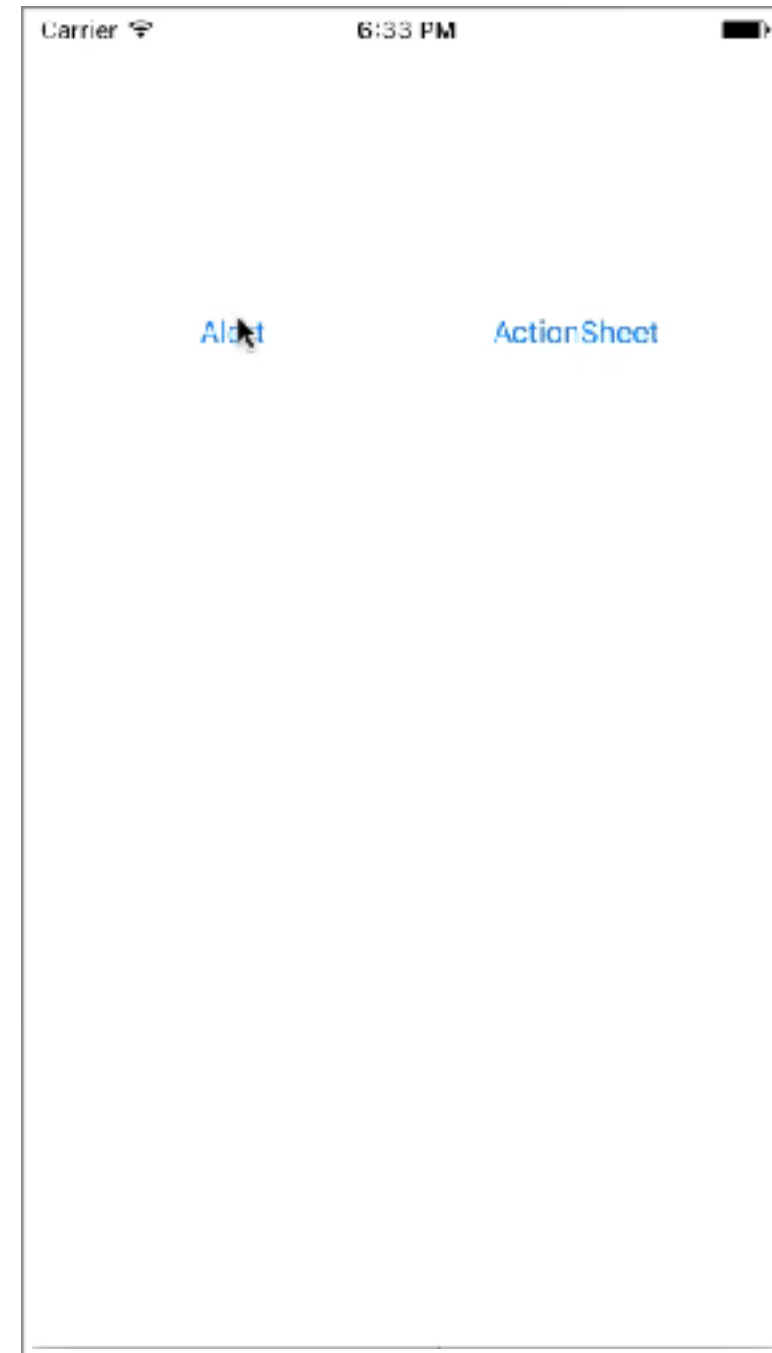
---

```
func btnAction(sender:UIButton) {  
    let alertVC = UIAlertController.init(title: "타이틀",  
                                         message: "알럿 메세지",  
                                         preferredStyle: .alert)  
  
    let okAction = UIAlertAction.init(title: "확인",  
                                       style: .default) { (action) in  
        //버튼 클릭시 실행 코드  
    }  
  
    alertVC.addAction(okAction)  
  
    self.present(alertVC, animated: true) {  
        //알럿 띄운 후 실행할 액션  
    }  
}
```

# Step 3. Exercise

---

- Alert // ActionSheet만들기
- 다양한 Action을 추가해보기





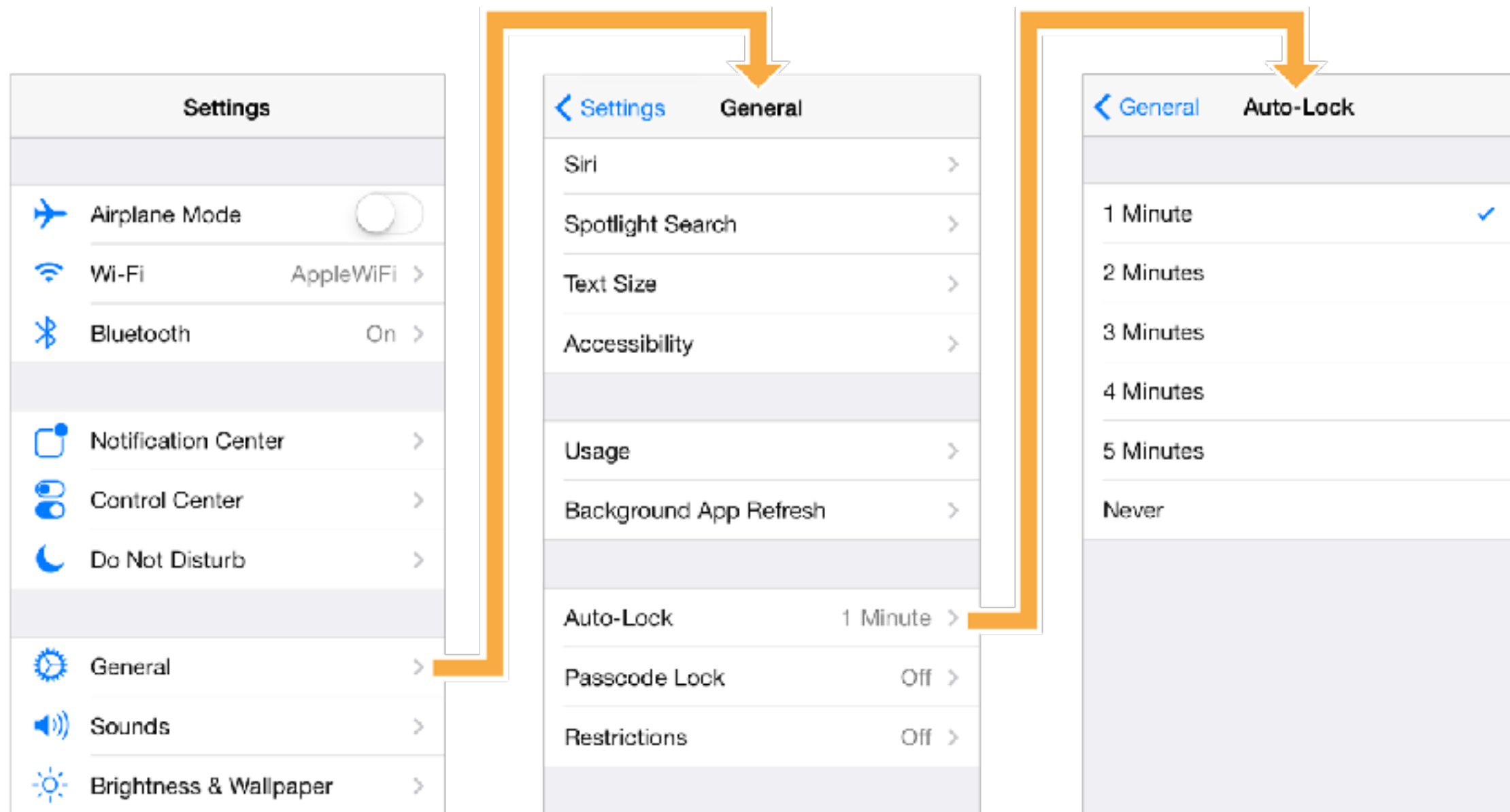
# Container View Controller

---

- View Controller의 Container역할을 하는 View Controller
- **View Controller 간의 구조를 잡는 역할을 한다.**
- 일반적으로 Root View를 가지고 있지 않고, View Controller를 Sub View Controller로 가지고 있다.
- 종류
  1. UINavigationController
  2. UITabBarController
  3. UISplitViewController

# 1. UINavigationController

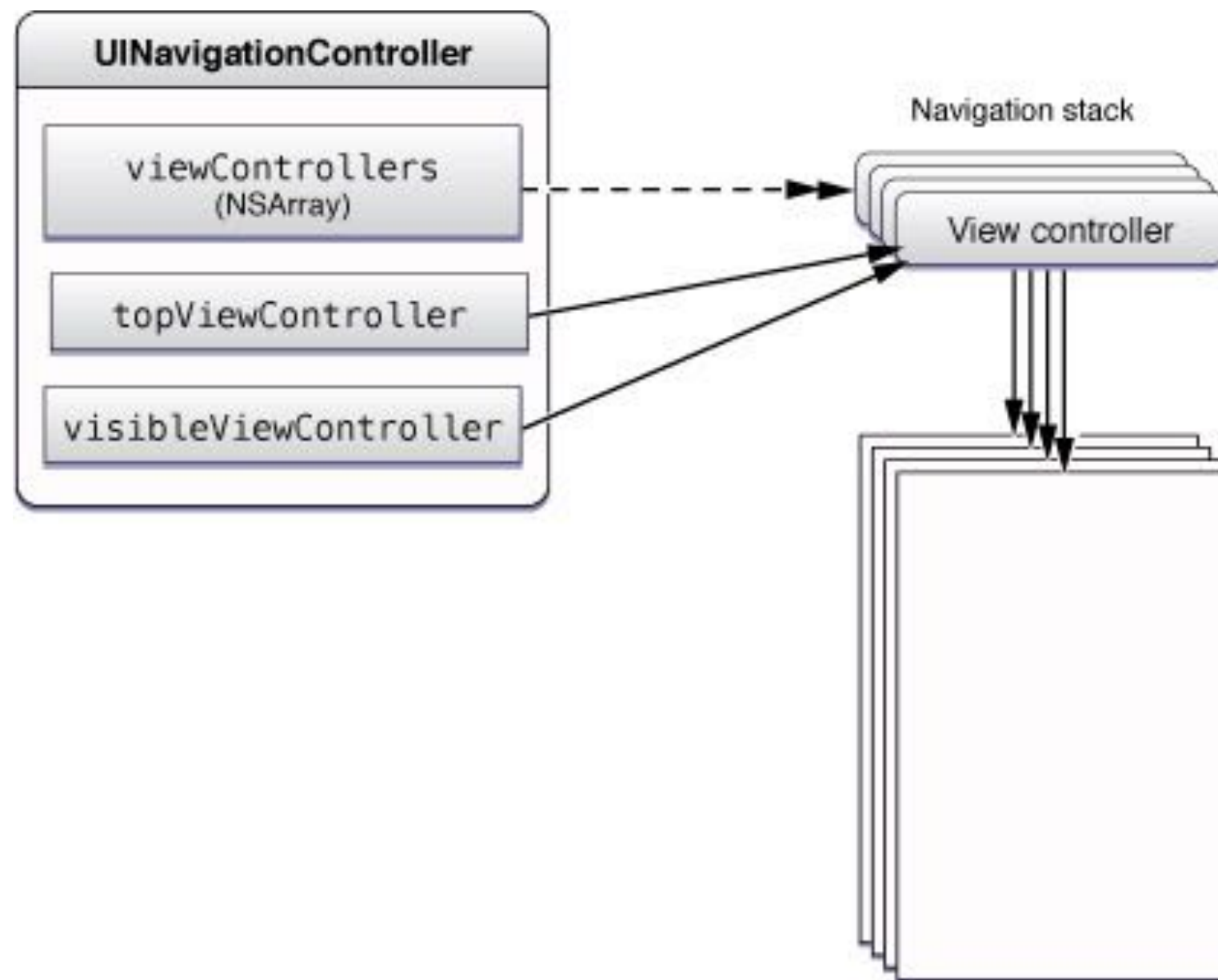
---



# NavigationController

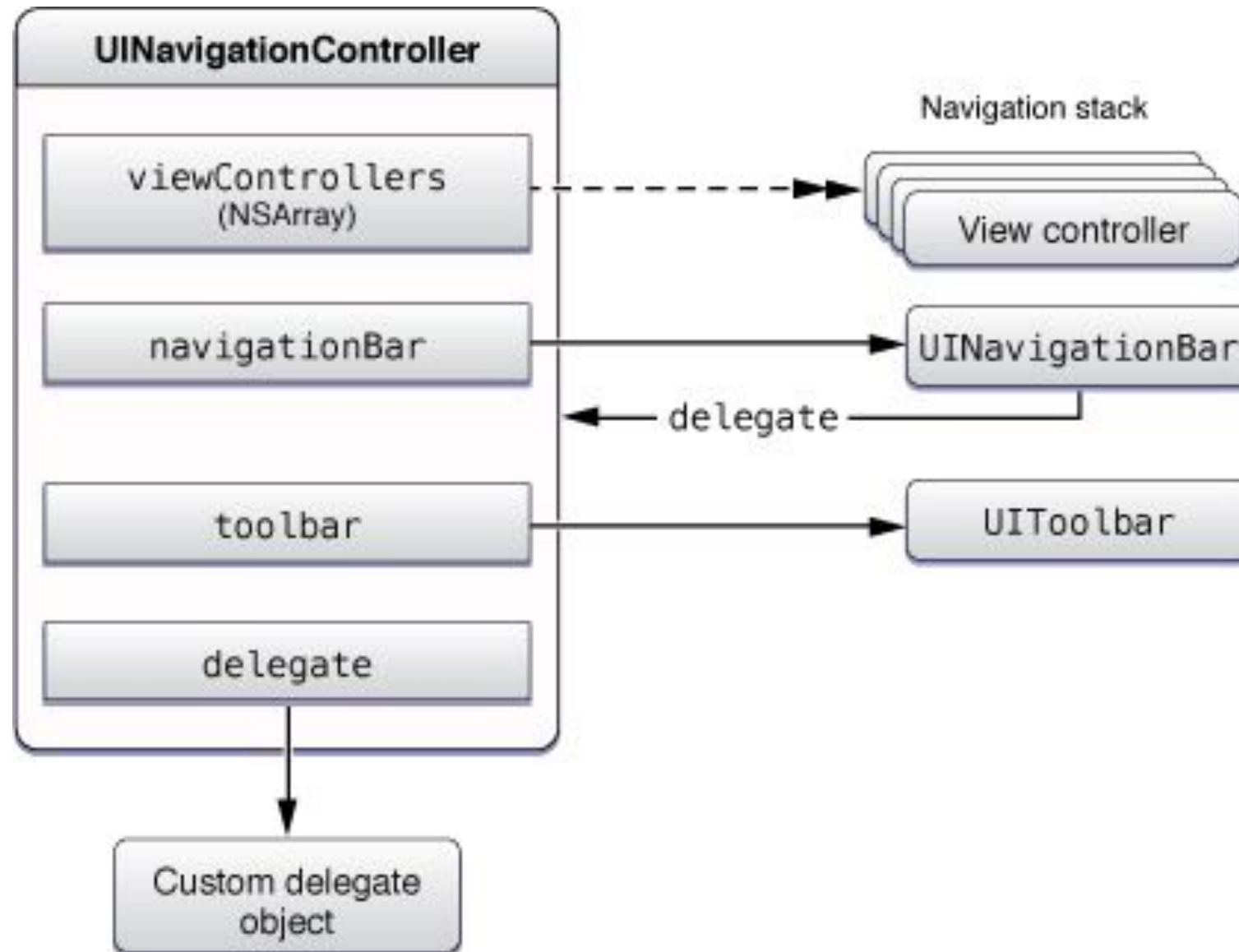
---

- UINavigationController class
- Navigation Controllers Manage Stacks of Other View Controllers



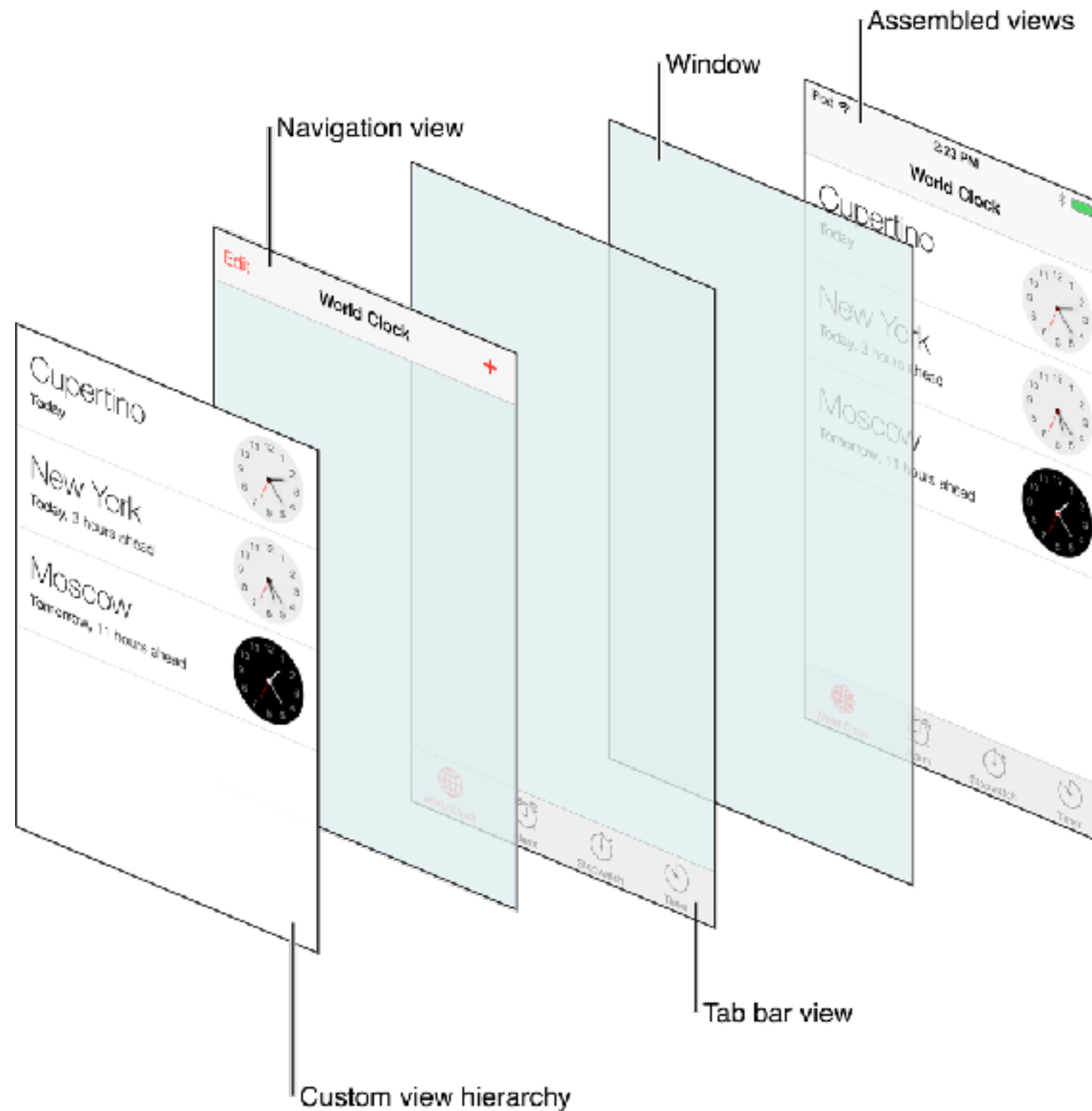
# NavigationController - 구조

---



# NavigationController Interface

---



# NavigationController 예제 - 스토리 보드

---

- ViewController선택
- Editor -> Embed in -> Navigation Controller
- Navigation Controller 지정

# NavigationController 실습

---

- Navigation Controller 기본 프로젝트 만들기
- Navigation Controller 화면 전환
- View Life Cycle 알아보기

# 화면 전환 실습

Carrier 5:22 PM

MY Login Page

ID :

PW :

[로그인](#)

[회원가입](#)

Carrier 5:23 PM

< 돌아가기

Signup Page

ID :

PW :

RE :

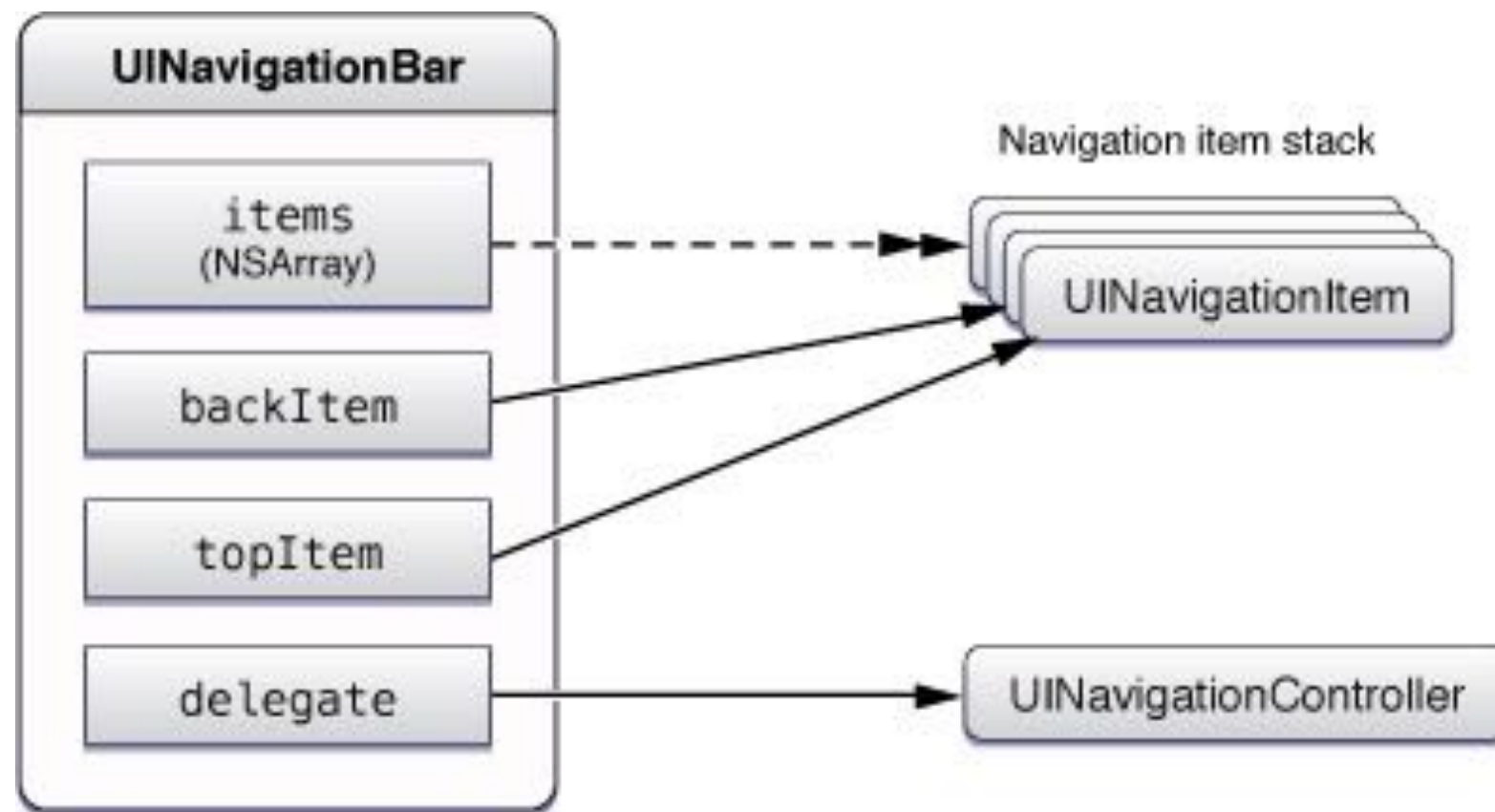
[회원가입](#)



# NavigationBar

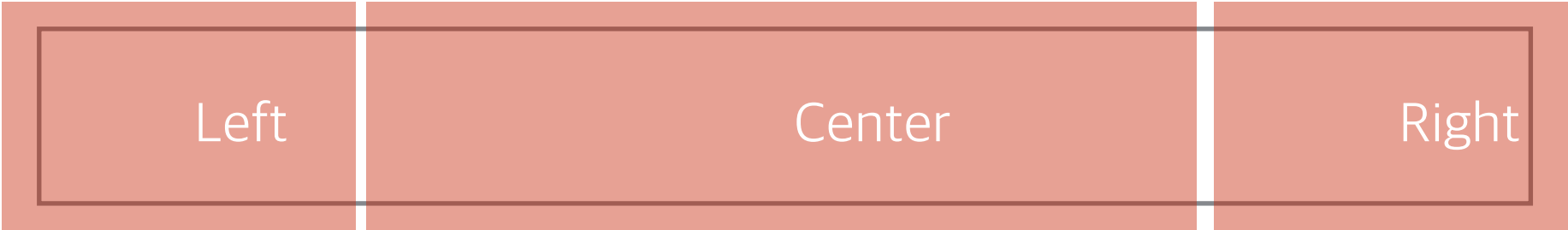
---

- 네비게이션 인터페이스를 관리 하는 뷰
- navigation bar의 외관은 customize할수 있다. 하지만
- frame, bounds, or alpha values는 절대 직접 바꿀수 없다.



# NavigationBar

---



Left	<code>backBarButtonItem leftBarButtonItem</code>
Center	<code>titleLabel</code>
Right	<code>rightBarButtonItem</code>

# Custom NavigationBar

---



Custom right bar button item



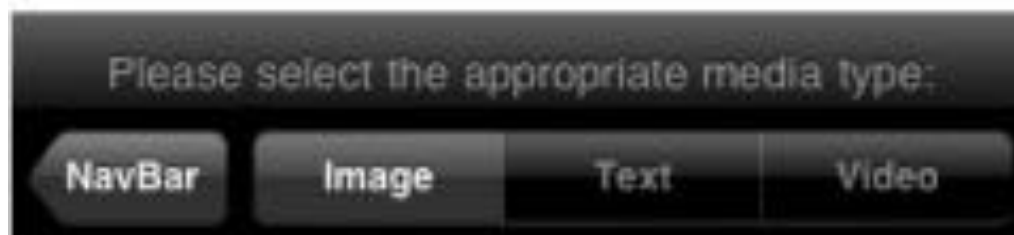
Custom right bar button with an image



Custom right bar button with a view



Custom title view



Custom title view and prompt

# Ref

---

- [goo.gl/2vQFRO](https://goo.gl/2vQFRO) //ViewController 관련 공식 문서
- <https://goo.gl/ntwhlm> //ViewController Catalog