

# UIViewController

# The Role of View Controllers

---

## 앱 구조의 뼈대

모든 앱에 반드시 하나 이상, 대부분 많은 수의 **ViewController** 로 구성

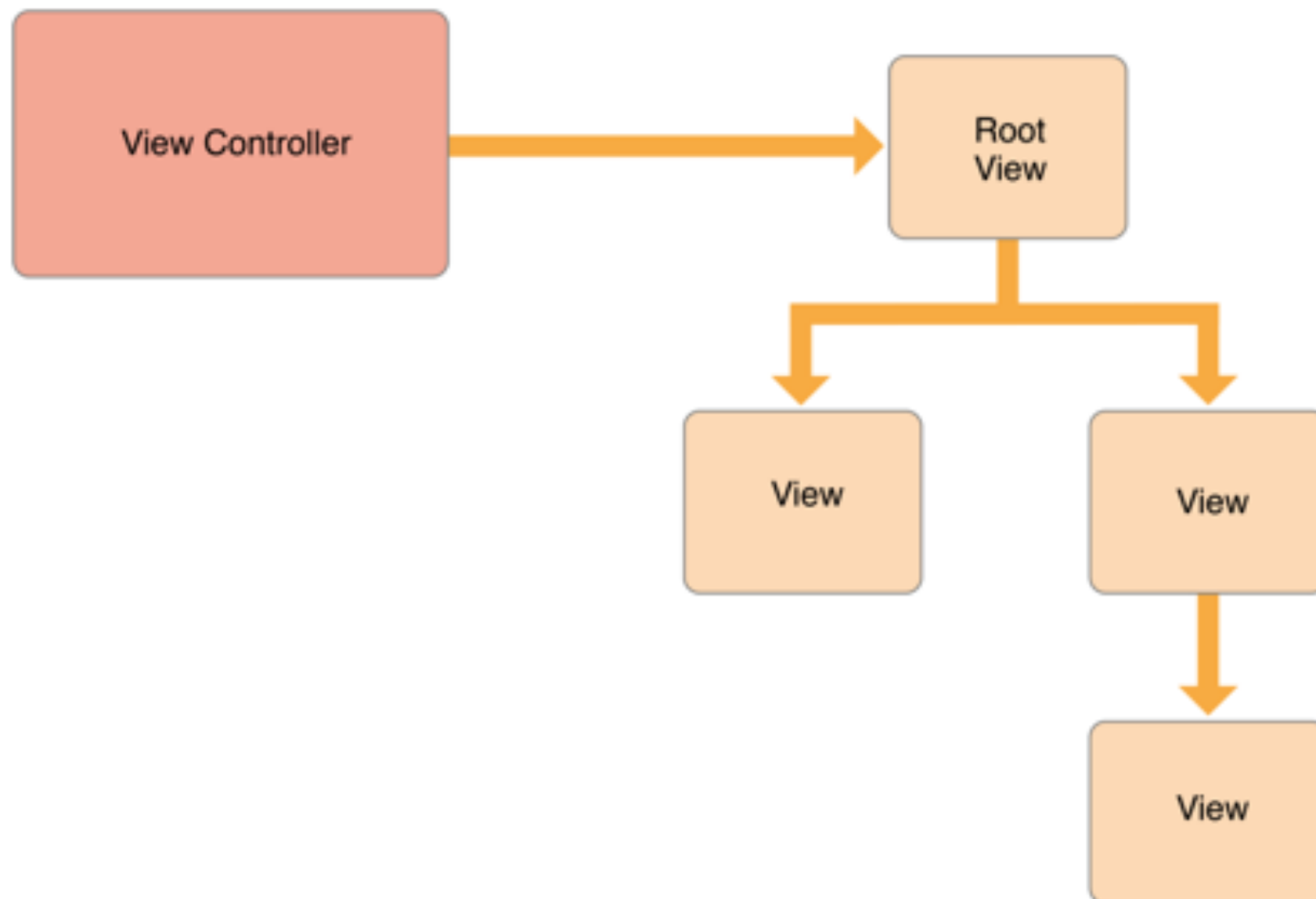
## 주요 역할

- View Management
- Data Marshaling
- User Interactions
- Resource Management
- Adaptivity

# View Management

가장 중요한 역할 - 뷰 계층 관리

모든 뷰컨트롤러마다 RootView를 지니며, 화면에 표시하기 위해서는 해당 RootView 계층에 속해야 함



# Two types of view controllers

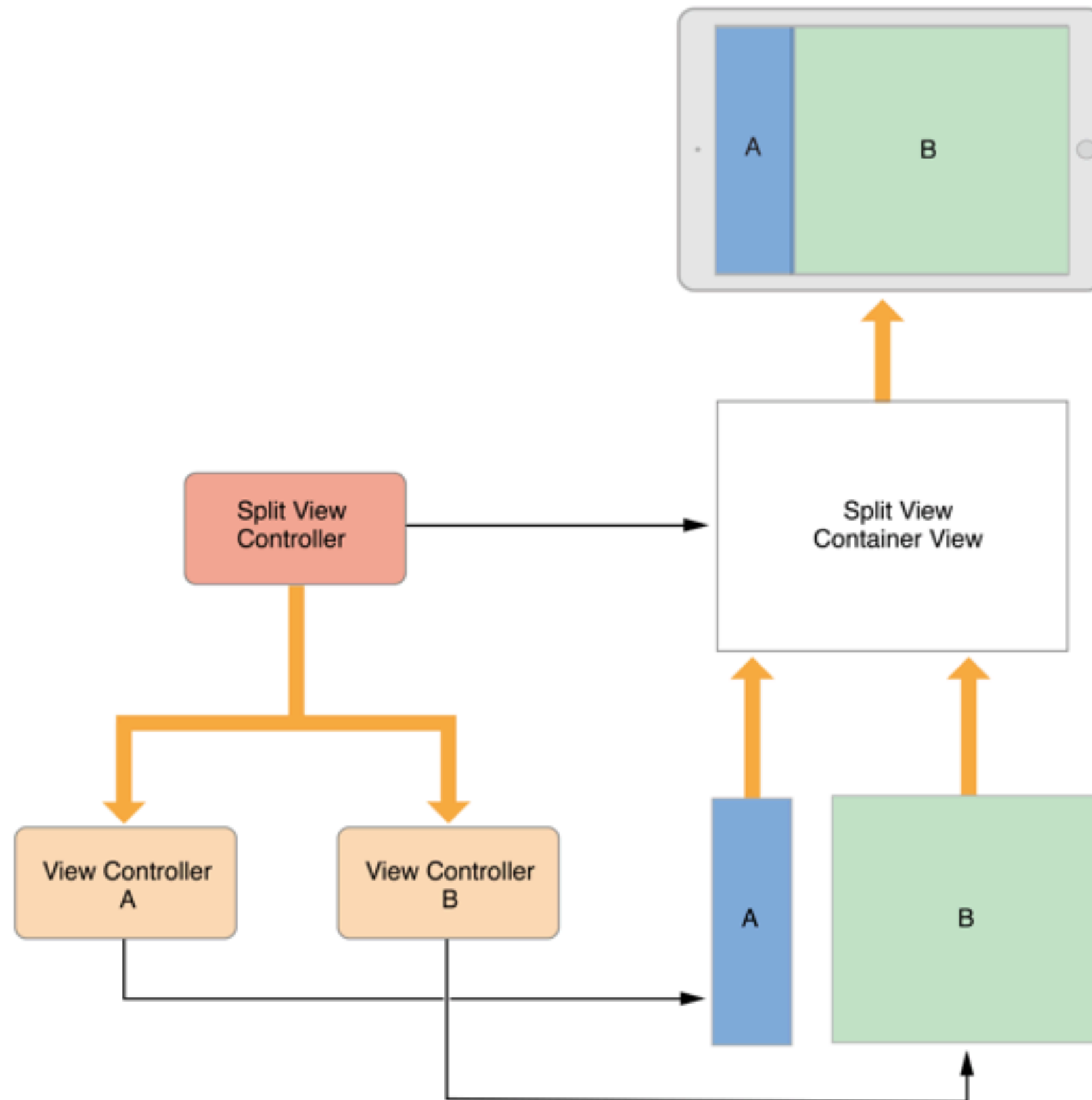
## Content View Controllers

- 모든 뷰를 단독으로 관리
- `UIViewController`, `UITableViewController`, `UICollectionViewController` 등

## Container View Controllers

- 자체 뷰 + 하나 이상의 자식 뷰 컨트롤러가 가진 루트뷰 관리
- 각 콘텐츠를 관리하는 것이 아닌 루트뷰만 관리하며 컨테이너 디자인에 따라 크기 조정
- `UINavigationController`, `UITabBarController`, `UIPageViewController` 등

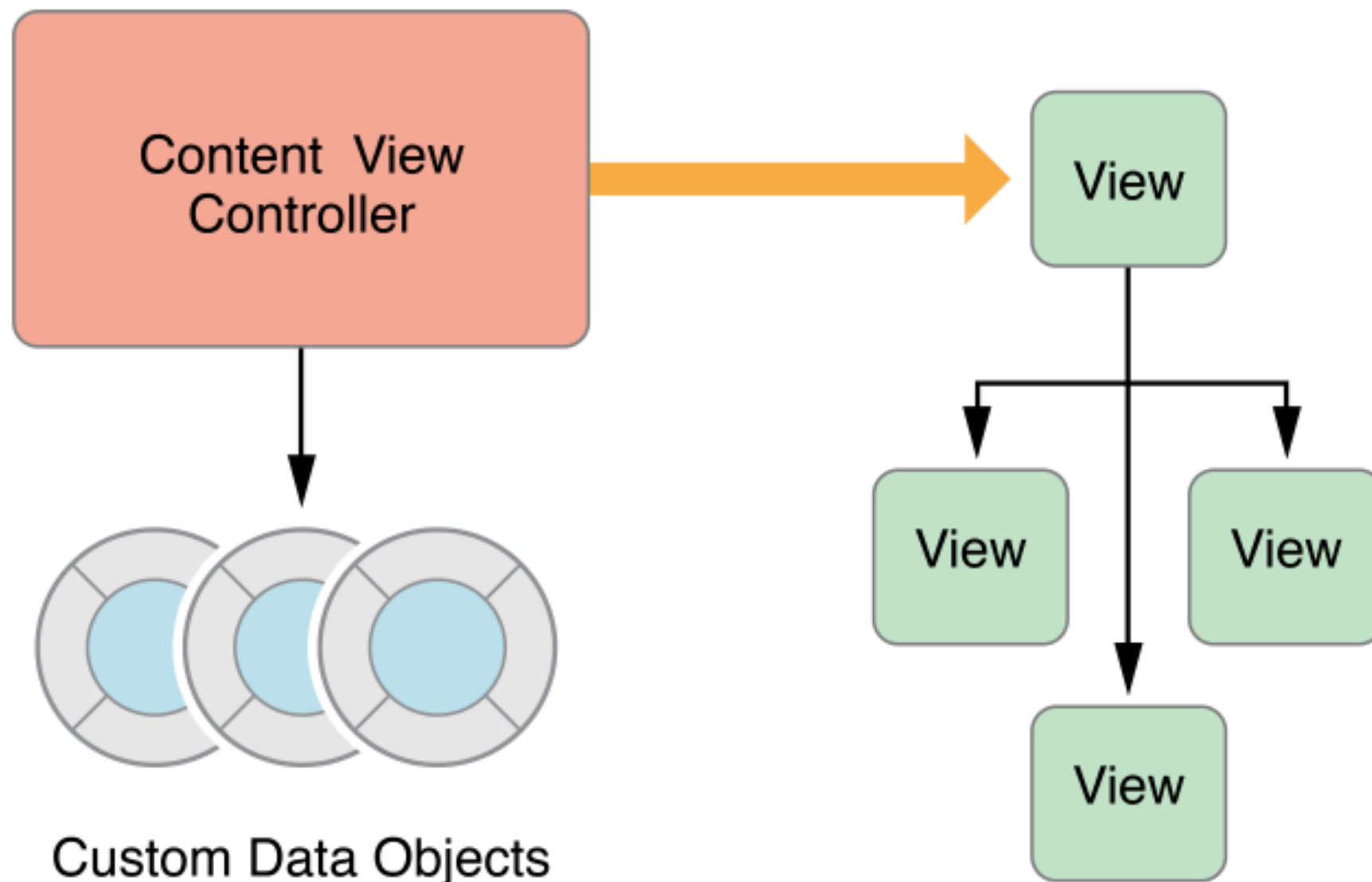
# Two types of view controllers



# Data Marshaling

## MVC (Model - View - Controller)

자신이 관리하는 View 와 Data 간 중개 역할



## 이벤트 처리

- 뷰컨트롤러는 Responder 객체 : 직접 이벤트를 받아 처리 가능하나 일반적으로 지양
- 뷰가 그 자신의 터치 이벤트를 연관된 객체에 action 메서드나 delegate 로 전달

# Resource Management

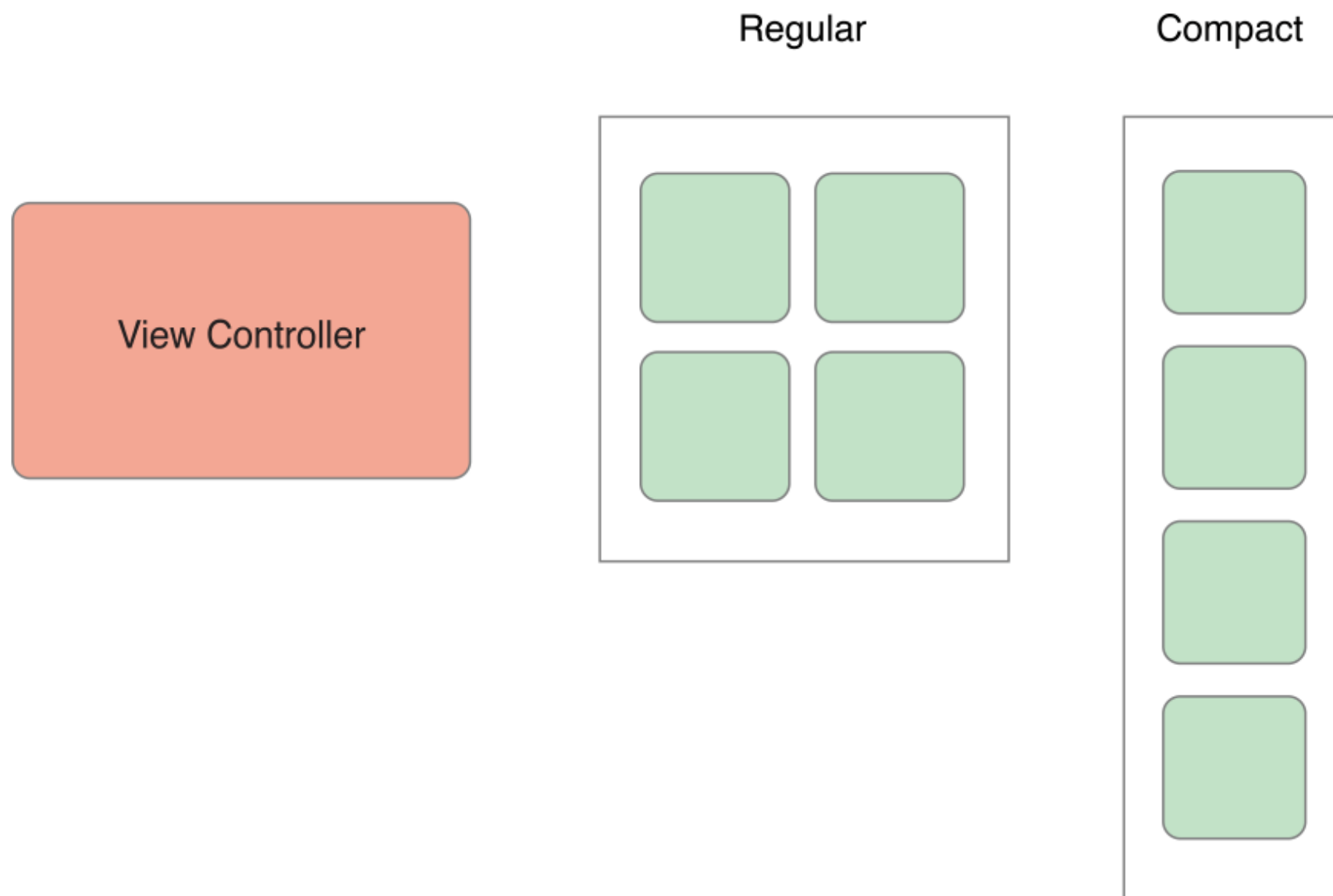
---

- 뷰컨트롤러가 생성한 모든 뷰와 객체들은 뷰컨트롤러의 책임
- 뷰컨트롤러의 LifeCycle 에 따라 생성되었다가 자동 소멸되기도 하지만  
ARC 개념에 맞게 관리 필요
- 메모리 부족 시 `didReceiveMemoryWarning` 메서드에서  
캐시 메모리 등 꼭 유지하지 않아도 되는 메모리들은 정리 필요



# Adaptivity

뷰컨트롤러는 뷰의 표현을 책임지고, 현재 환경에 적절한 방법으로 적용되도록 할 책임을 가짐

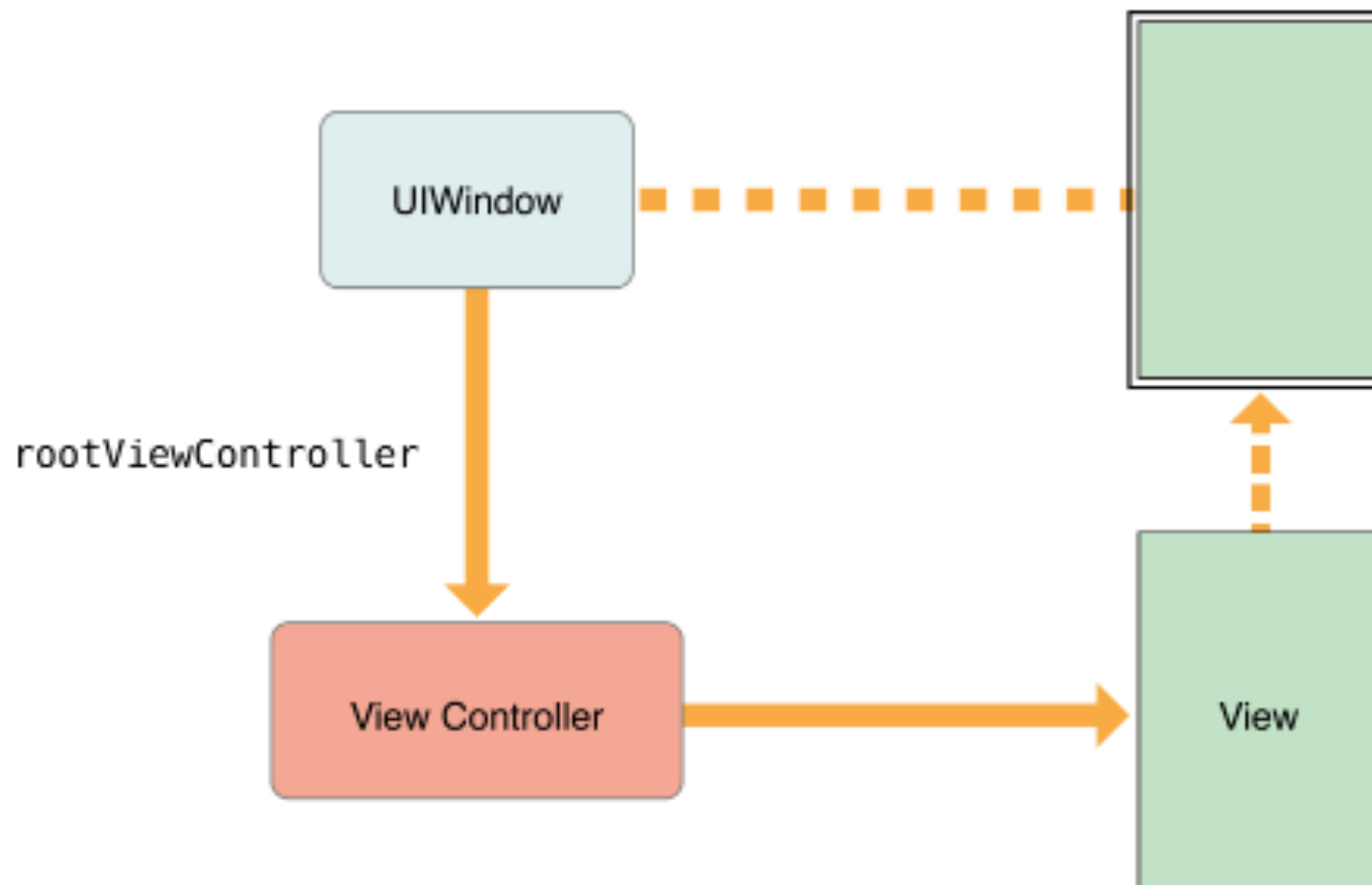


# The View Controller Hierarchy

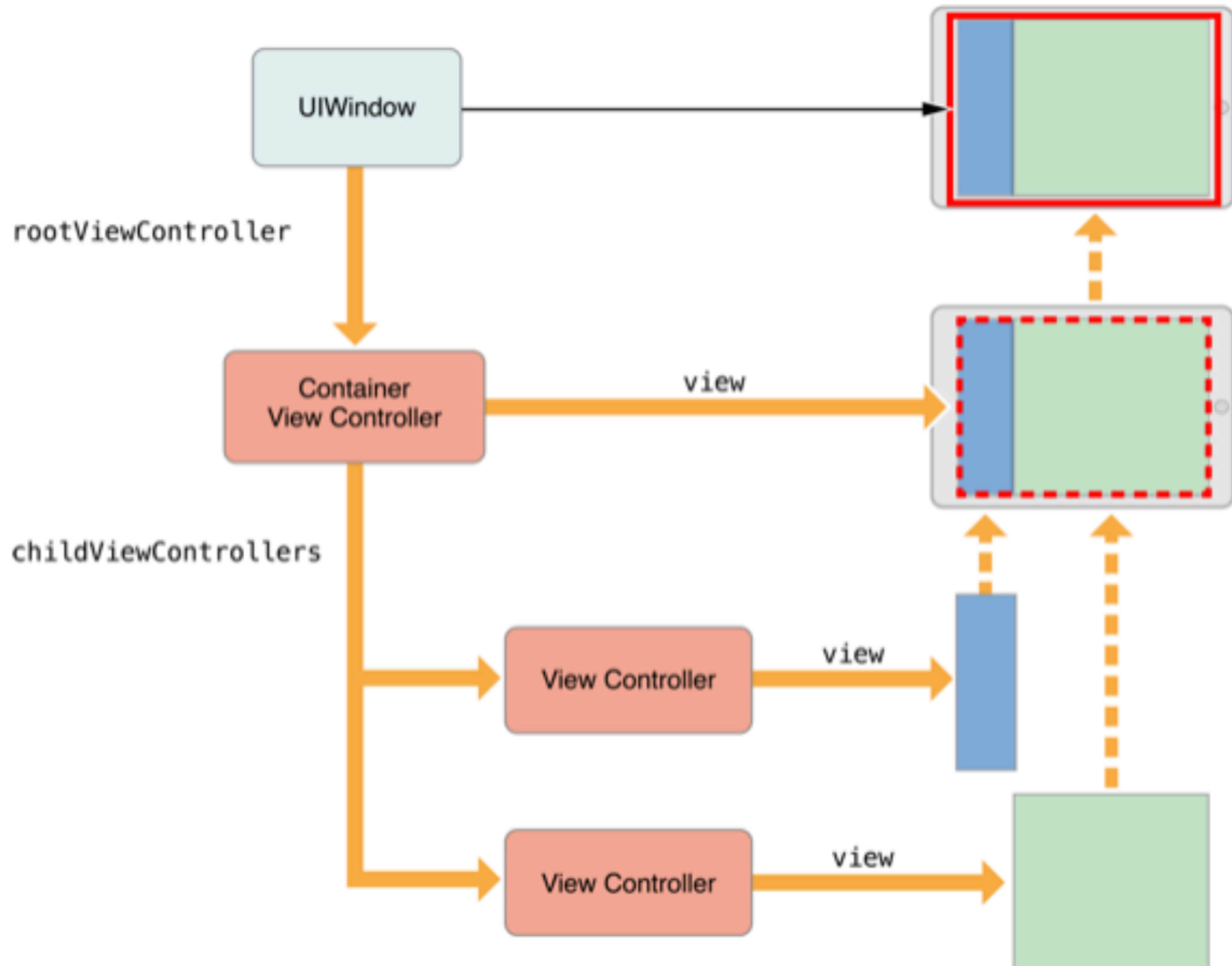
# The Root View Controller

UIWindow 는 그 자체로는 유저에게 보여지는 콘텐츠를 가지지 못함

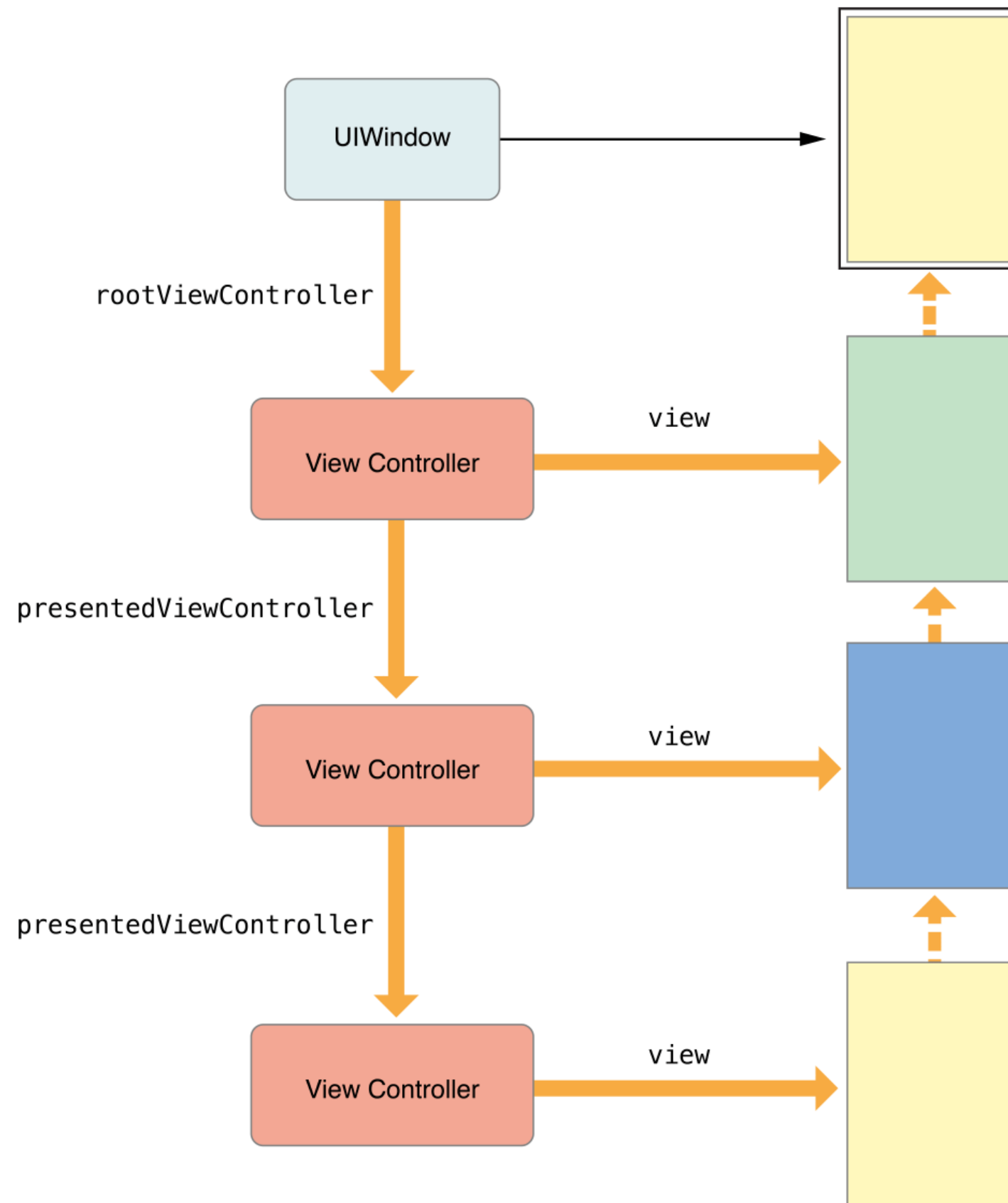
Window는 정확히 하나의 Root View Controller 를 가지는데 이것을 통해 콘텐츠를 표현



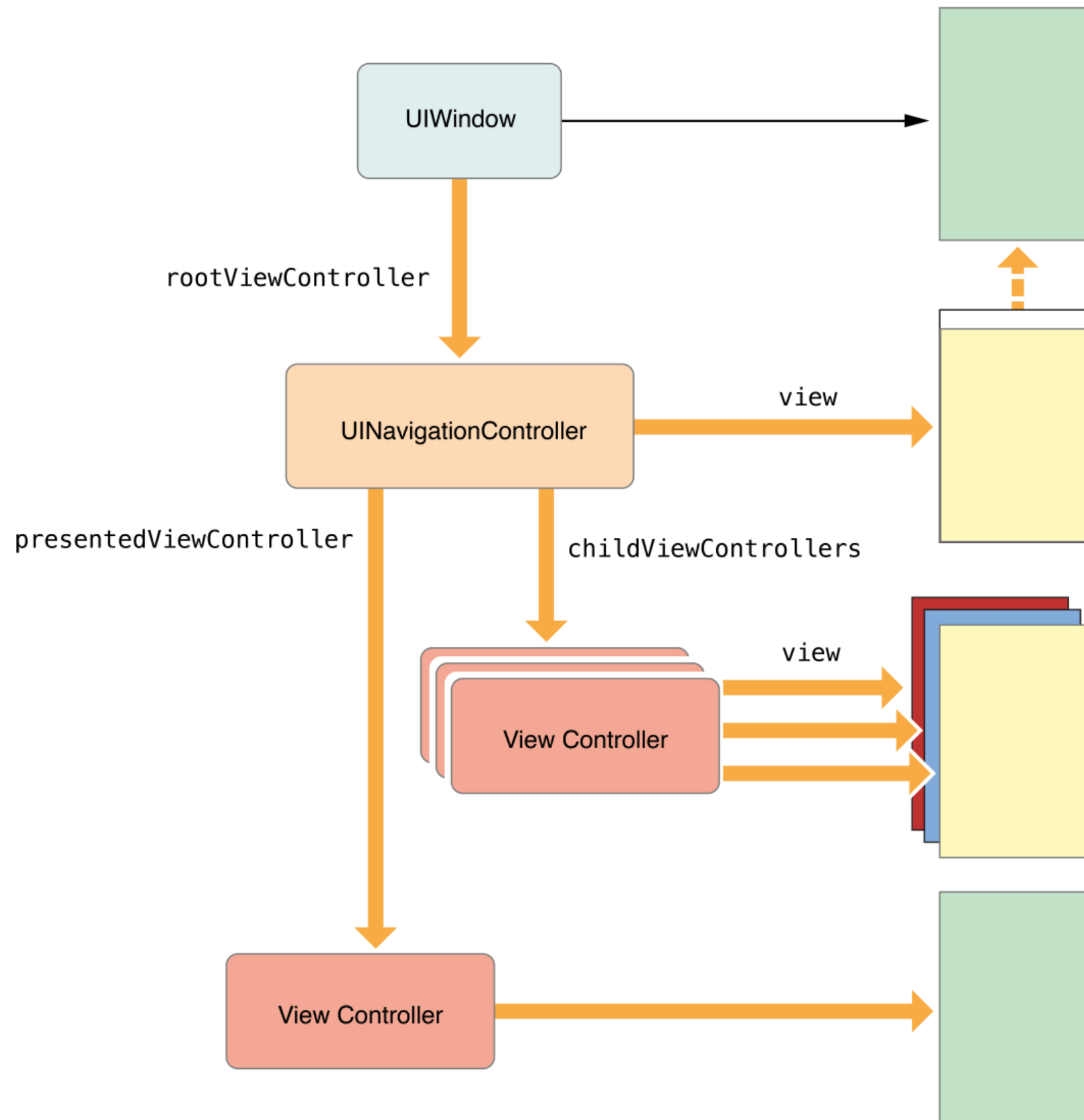
# Container View Controllers



# Presented View Controllers



# A container & a presented View Controller



# View Controller Life Cycle

# ViewController Life Cycle

