iOS View Controller Life Cycle

Introduction:

It’s important to understand Apps Life cycle and View Controller life Cycle before start to develop iOS Apps.So today i will discuss **View controllers** life cycle and few basic of View controller .

What you will learn from this tutorial :

What is View Controller .

Why View Controller important.

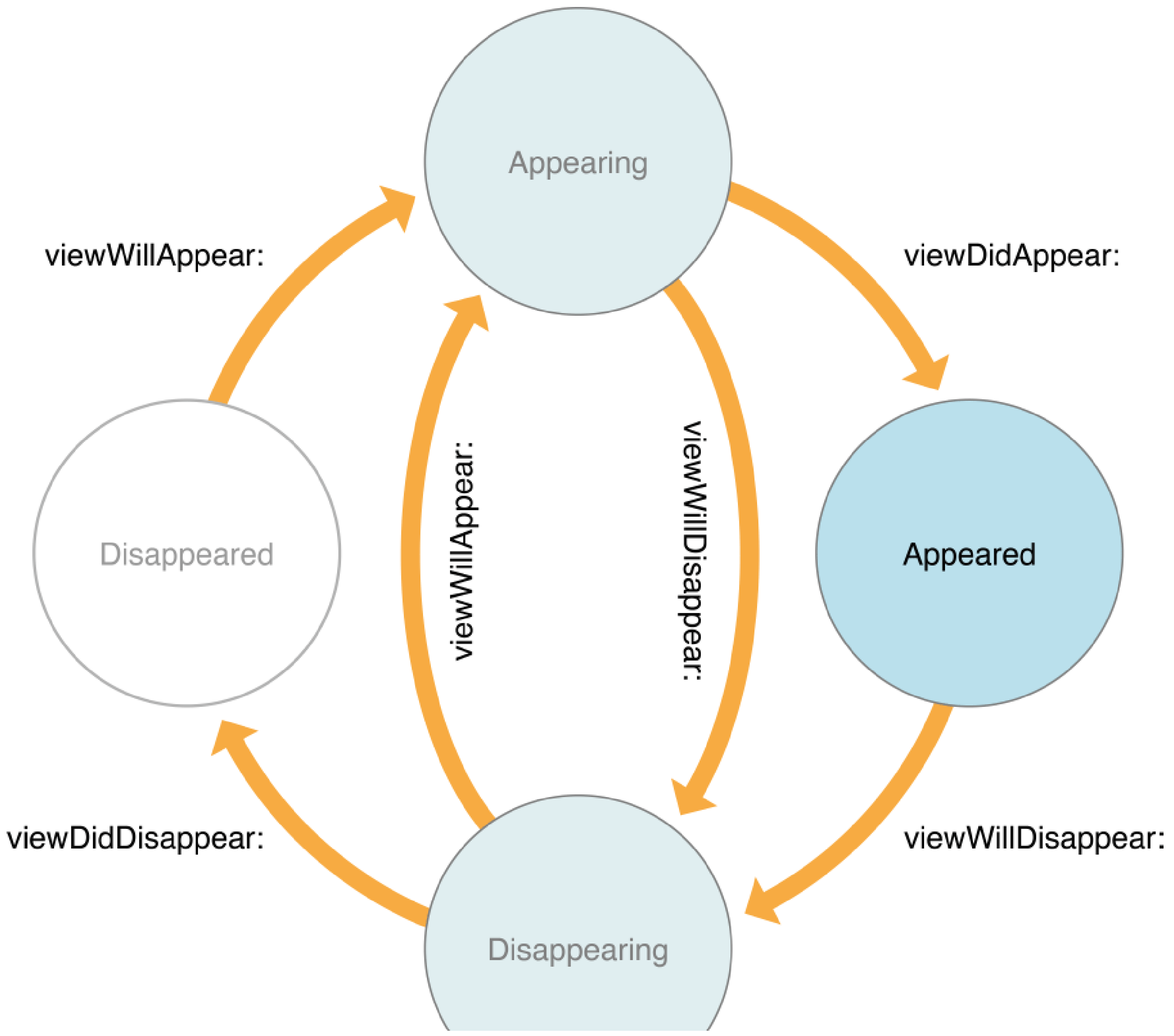
How view Controller work .

View controller life cycle.

There is three way to create UI (Which attached with a viewController class).

1. From the .xib
2. From Code
3. By Storyboard

View controller Life Cycle :



loadView

This method use when view Controller create from code .Its good not to do anything on this method .If view Controller made from .xib or storyboard.

What Do in View Load : loadView( ) is a method managed by the viewController. The viewController calls it when its current view is nil. loadView( ) basically takes a view (that you create) and sets it to the viewController's view (superview).

viewDidLoad:

This Method is loaded once in view controller life cycle .Its Called When all the view are loaded .You Can do Some common task in this method :

1.Network call which need Once.

2.User Interface

3.Others Task Those are Need to do Once

> Note: This Method Call before the bound are defined and rotation happen.So its Risky to work view size in this method.

viewWillAppear:

This Method is called every time before the view are visible to and before any animation are configured .In this method view has bound but orientation not set yet.You can override this method to perform custom tasks associated with displaying the view such as to hide fields or disable actions before the view becomes visible.

viewWillLayoutSubviews:

It don't do Nothing by default. When a view'€   s bounds change, the view adjusts the position of its subviews. View controller can override this method to make changes before the view lays out its subviews.

viewDidLayoutSubviews:

This method is called after the viewController has been adjust to its subview following a change on its bound.Add code here if you want to make change to subviews after they have been set.

viewDidAppear:

This Method is called after the view present on the screen. Usually save data to core data or start animation or start playing a video or a sound, or to start collecting data from the network This type of task good for this method.

viewWillDisappear:

This method called before the view are remove from the view hierarchy.The View are Still on view hierarchy but not removed yet . any unload animations haven't been configured yet. Add code here to handle timers, hide the keyboard, cancel network requests, revert any changes to the parent UI. Also, this is an ideal place to save state.

viewDidDisappear:

This method is called after the VC's view has been removed from the view hierarchy. Use this method to stop listening for notifications or device sensors.

deinit:

Before a view controller is removed from memory, it gets deinitialized. You usually override deinit() to clean resources that the view controller has allocated that are not freed by ARC. Keep in mind that just because a VC is no longer visible, doesn't mean that it has been deallocated. Container view controllers such as NavigationController can keep their VCs available in memory. Keep in mind that even though a VC is off screen, if it is still in memory, it still works normally and can receive notifications.

didReceiveMemoryWarning()

When memory starts to fill up, iOS does not automatically move data from memory to its limited hard disk space. It does, however, issue this warning and YOU (I mean YOU) are responsible for clearing some objects out of memory. Be aware that if the memory of your app goes over a certain threshold, iOS will shutdown your app. Unfortunately, this will look like a crash to the end user.

viewWillTransition(to:with:)

When the interface orientation changes, UIKit calls this method on the window’s root view controller before the size changes are about to be made. The root view controller then notifies its child view controllers, propagating the message throughout the view controller hierarchy. The parameter to contains the new CGSize size of the container’s view and the parameter with contains a UIViewControllerTransitionCoordinator coordinator, an enum that describes the new orientation.