

iOS Dev Accelerator

Week 7 Day 4

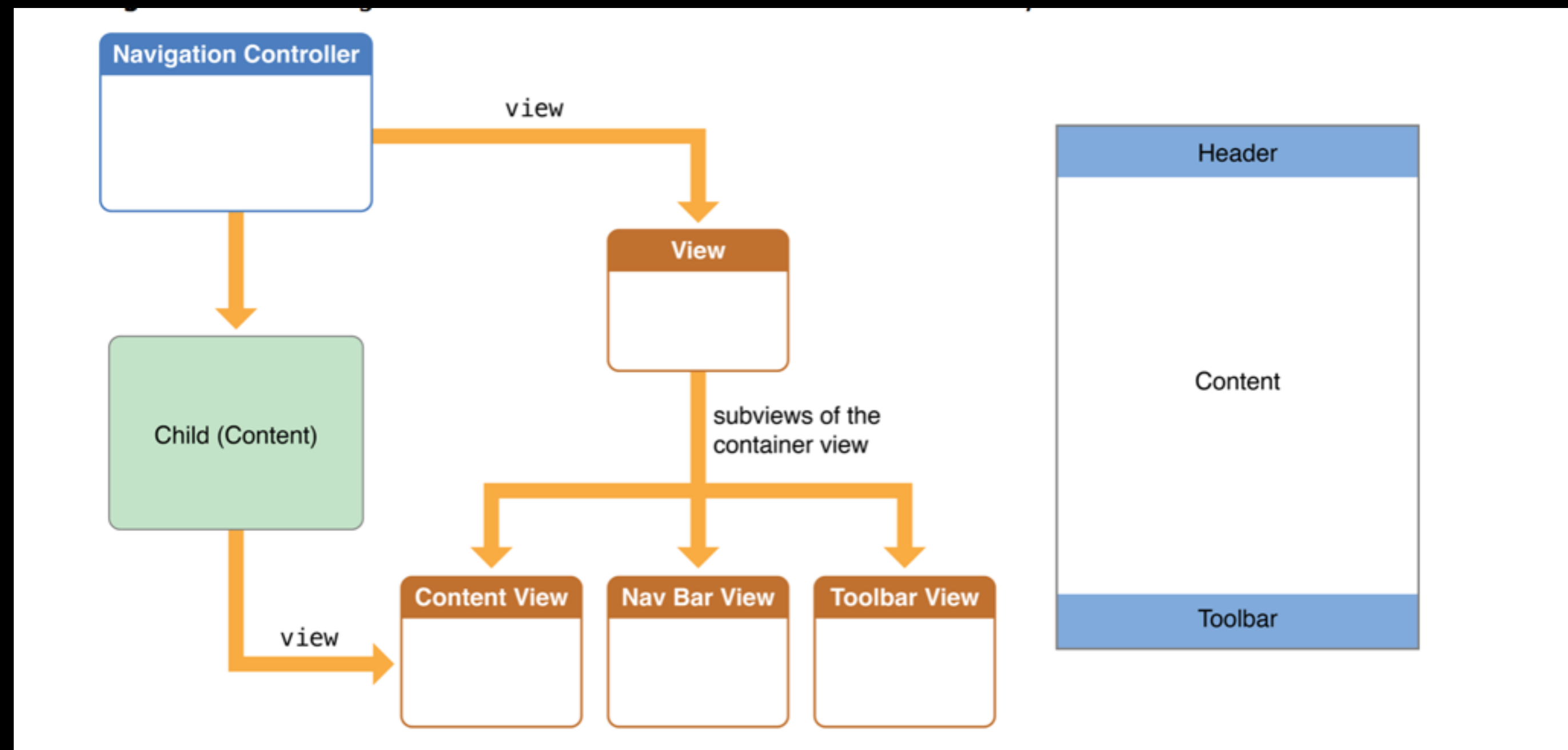
- Container View Controllers
- Notification Center
- Trello

Container View Controllers

- Most of the view controllers you have been working with so far are considered 'Content View Controllers'
- Container View Controllers are similar to Content View Controllers, except they manage parent-child relationships between the Container VC, the parent, and its Content VC(s), the children.
- The Container VC is in charge of displaying its children VC's views.

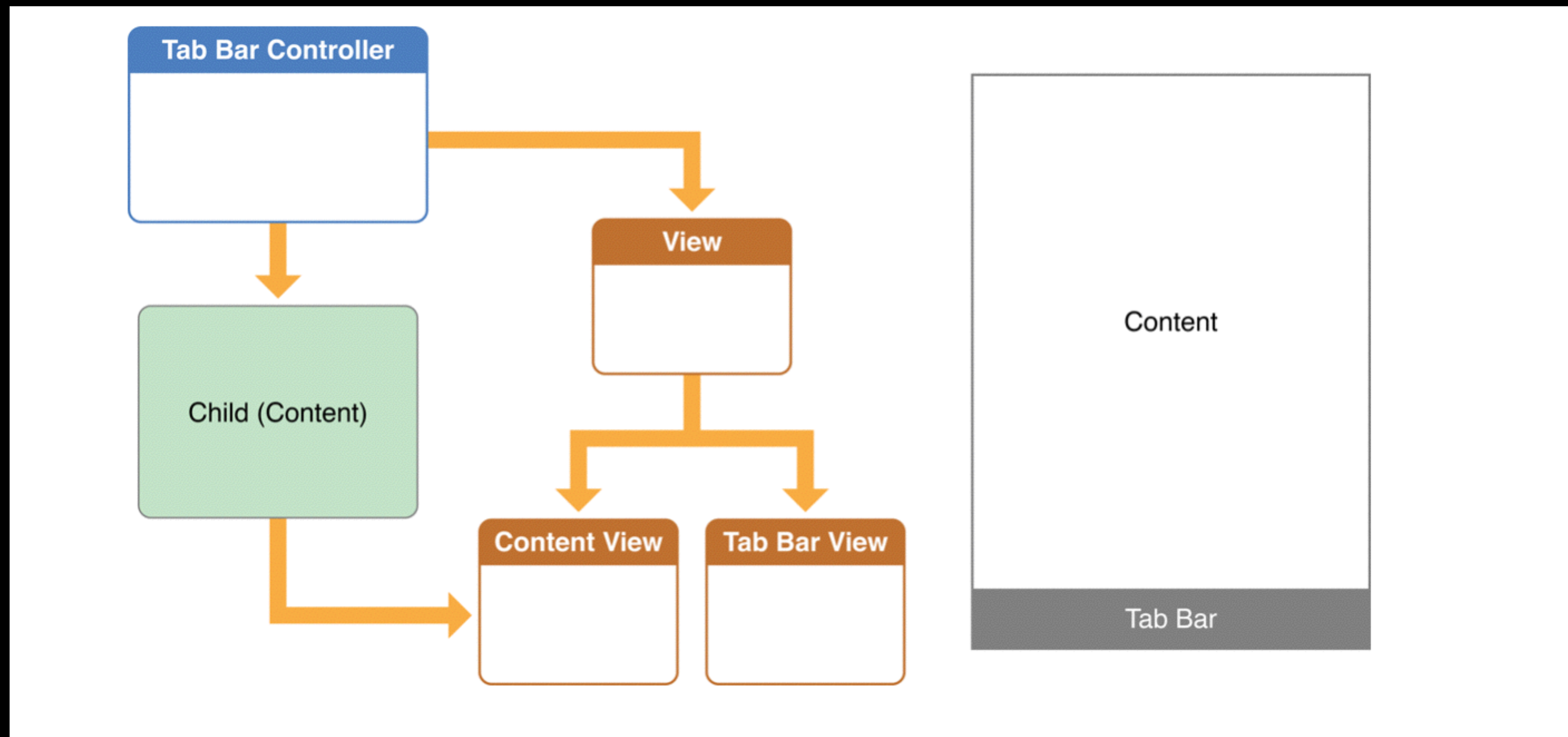
Container View Controllers

- An example of a Container View Controller you have used is the Navigation Controller:



Container View Controllers

- Same Idea with a Tab Bar Controller:



Container View Controllers

- “The goal of implementing a container is to be able to add another view controller’s view as a subview in your container’s view hierarchy”
- When a new VC is added on screen, you can ensure the right events (viewDidLoad, viewWillAppear, etc) are given to all VC’s by associating the new VC as a child of the container VC.

Getting your child on screen

```
func setupChildVC() {  
    //create a new VC  
    var childVC = UIViewController()  
    //tell this VC that we are adding a child VC  
    self.addChildViewController(childVC)  
    //set the child VC's view's frame, in this case it will  
        completely cover the container VC  
    childVC.view.frame = self.view.frame  
    //add the child view to the parent view  
    self.view.addSubview(childVC.view)  
    //notify the child vc he is now on screen  
    childVC.didMoveToParentViewController(self)  
}
```


Taking the child off screen

```
childVC.willMoveToParentViewController(nil)  
childVC.view.removeFromSuperview()  
childVC.removeFromParentViewController()
```

Transitioning your Child VCs

```
//being process of removing old one and adding new one
childVC.willMoveToParentViewController(nil)
self.addChildViewController(nextChildVC)

//set the new child's frame to be off screen to the left, and create an end frame
//that is off the screen to the right
nextChildVC.view.frame = CGRectMake(-700, 0, self.view.frame.width, self.view.
    frame.height)
var endFrame = CGRectMake(700, 0, childVC.view.frame.width, childVC.view.frame.
    height)

//call the transition method on our container view controller
self.transitionFromViewController(childVC, toViewController: nextChildVC,
    duration: 0.5, options: UIViewAnimationOptions.TransitionCrossDissolve,
    animations: { () -> Void in
        //give our new child a frame equal to our containers view, putting him fully
        //on screen. set our old child's view to the end frame so it slides off to
        //the right
        nextChildVC.view.frame = self.view.frame
        childVC.view.frame = endFrame
    }) { (success) -> Void in

        //finish the operation|
        childVC.removeFromParentViewController()
        nextChildVC.didMoveToParentViewController(self)
    }
}
```

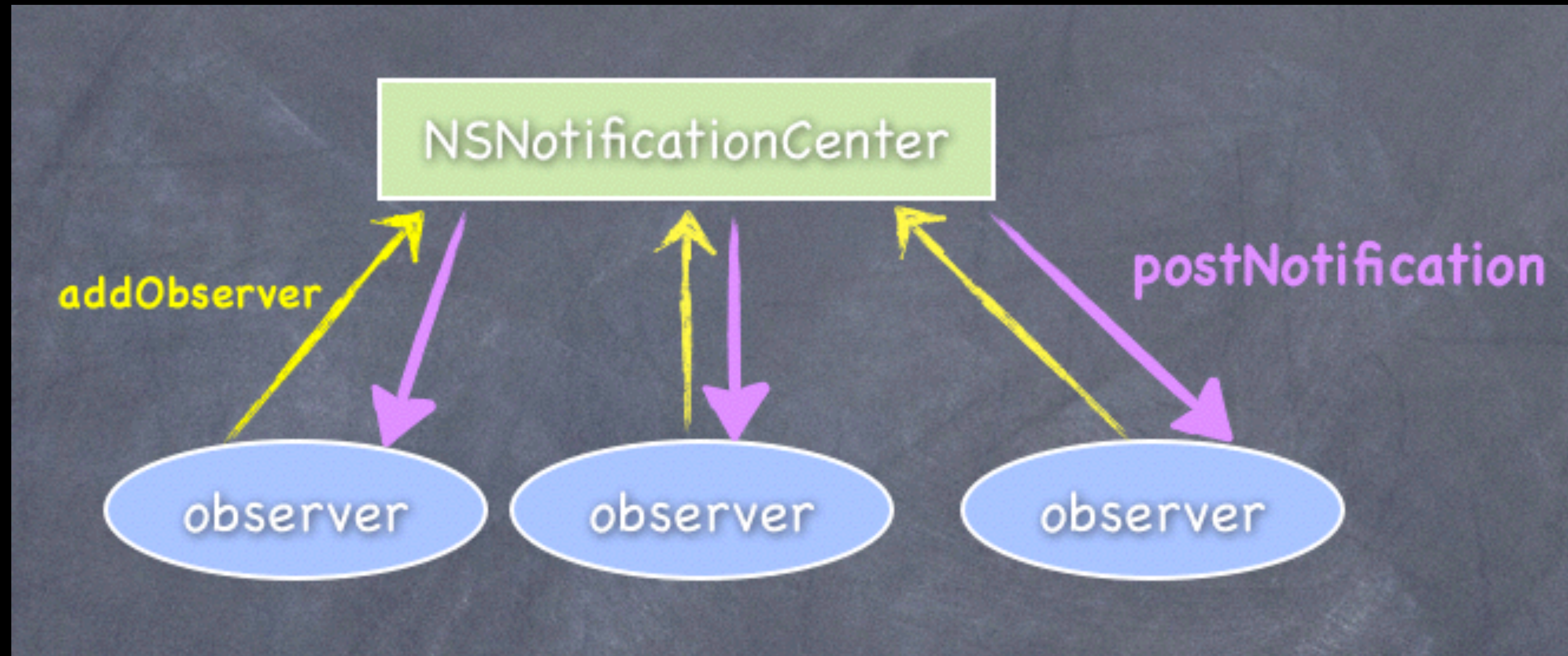

KVO (Key-Value Observing)

- Key-value observing provides a mechanism that allows objects to be notified of changes to specific properties of other objects.
- A controller object typically observes properties of model objects
- A view object typically observes properties of model objects through a controller

3 Steps of KVO in Cocoa

1. Register as an observer `addObserver:forKeyPath:options:context:`
2. Implement `observeValueForKeyPath:ofObject:change:context:`
3. Remove your observer when finished `removeObserver:forKeyPath:`

NSNotificationCenter



NSNotificationCenter Prerequisites

1. Name of the notification
2. Optionally listen to a specific object
3. Optionally specify a callback queue
4. Code inside a closure to run every time notification fires

```
NSNotificationCenter.defaultCenter().addObserverForName(  
    UITextFieldTextDidChangeNotification,  
    object: nil,  
    queue: NSOperationQueue.mainQueue() ) {  
        (note) in  
            // any code written here will run every time notification fires  
    }
```


How To Observe Everything

(don't do this in production)

```
NSNotificationCenter.defaultCenter().addObserverForName(nil, object: nil, queue: nil) {  
    (note) -> Void in  
    println("Notification Received: \(note.name)")  
}
```