Modally Presenting View Controllers

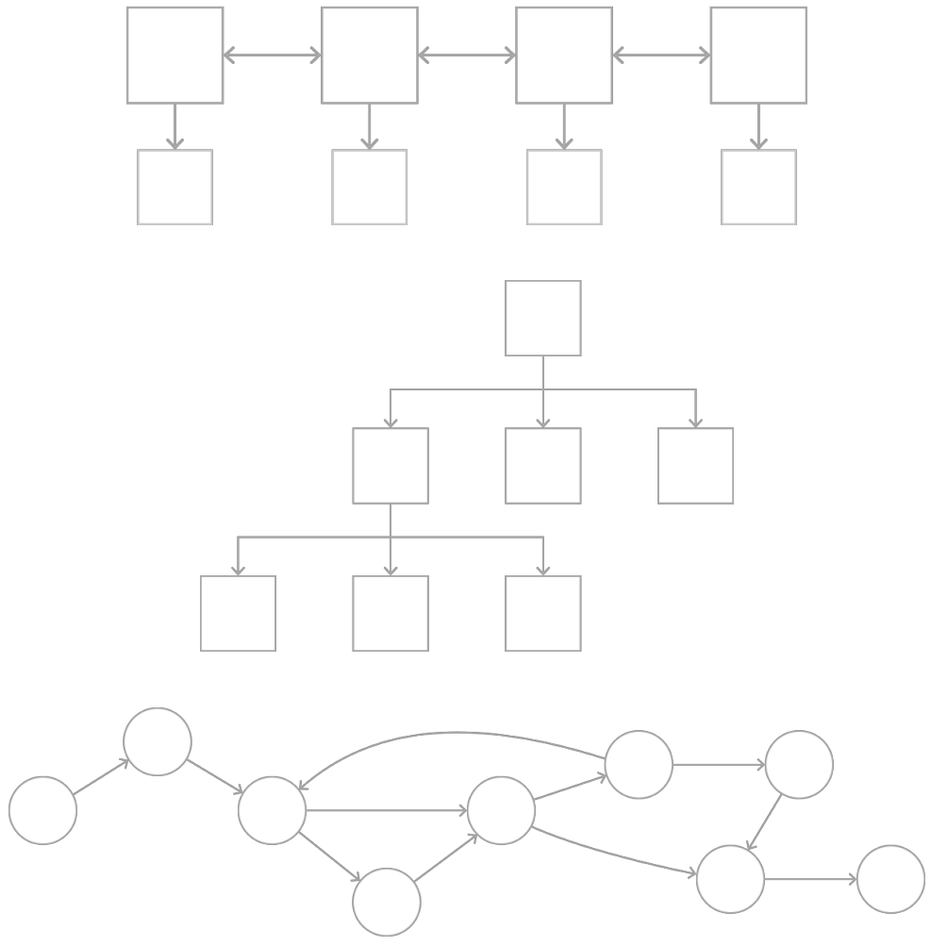
Mobile Computing - iOS

Objectives

- Students will be able to:
    - explain how to present alerts and view controllers in storyboard using segues
    - define modality, and explain the difference between a modal and non-modal presentation
    - describe iOS's presentation and transition styles, and how to set them in storyboard

Overview

- There are several ways to navigate through screens of content
    - flat navigation, to switch between multiple screens (tab bar controllers)
    - hierarchical navigation, to dive into data (navigation controllers)
    - experience-driven/content-driven (e.g., in a game or book)
- These can be combined, as well. For instance, one tab in a tabbed app might present hierarchical data



Displaying a View Controller

- A view controller becomes visible on screen when it is either:
  - **embedded** in a container view controller, allowing the container view controller's view to remain visible
  - **presented**, when it completely covers the current view controller

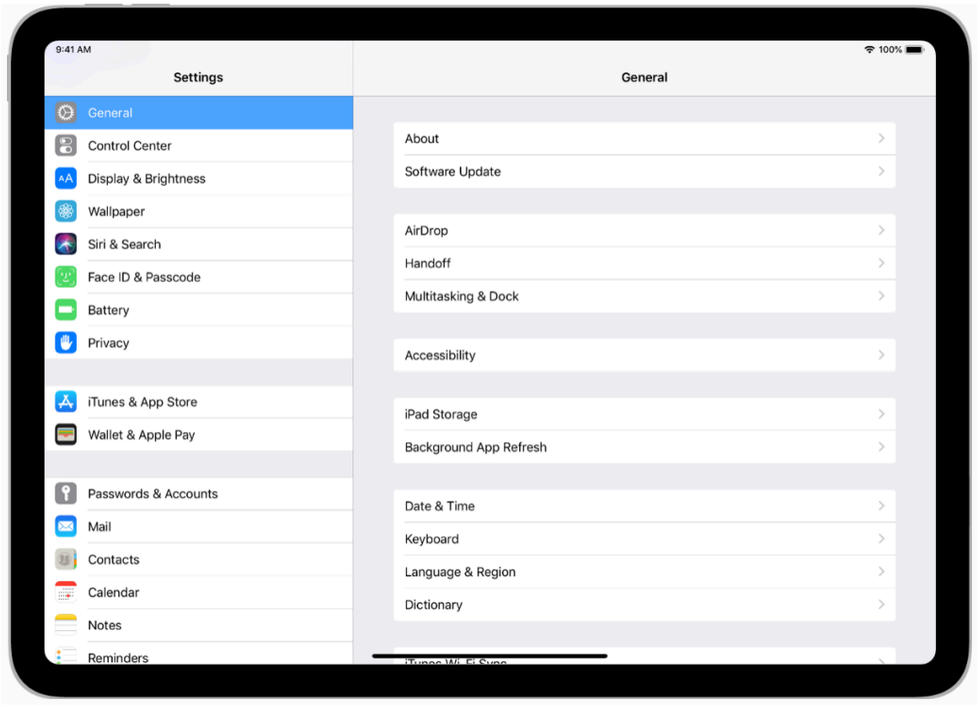
The Show Must Go On

- When a container controller, in the course of its business, displays another view controller (VC), it is said to show that VC.

- When a navigation controller shows a VC it pushes it onto the stack. Since the VC is embedded in the navigation controller, the navigation controller's view (aka the navigation bar and (possibly) tool bar) remain visible.

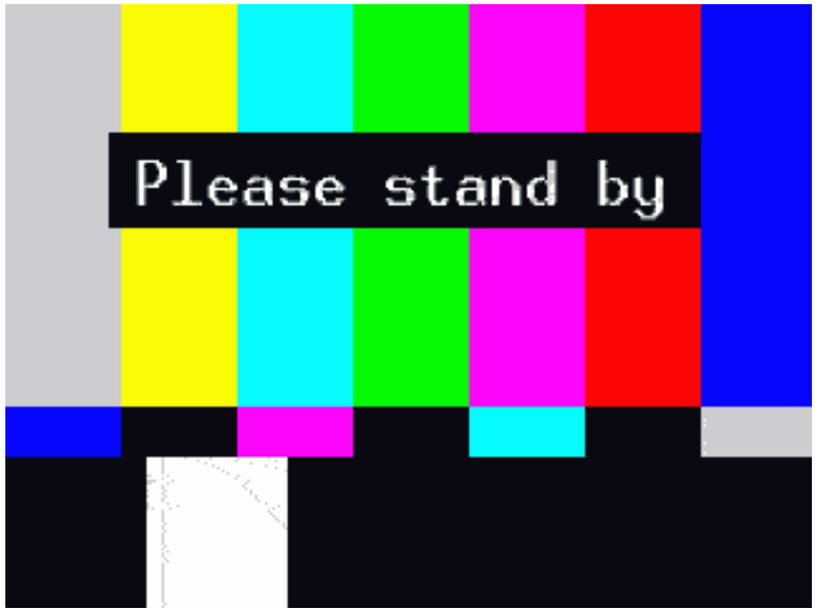
- When a tab bar controller shows a VC, it just sets its selectedIndex to the index of the VC in the viewControllers array (remember that?). The tab bar controller's view remains visible (hence the tab bar).

- A split view controller shows a VC by placing it on the detail side, in response to interaction on the master side. Here, too, the entire split view remains visible.



We Interrupt This Broadcast: Modality

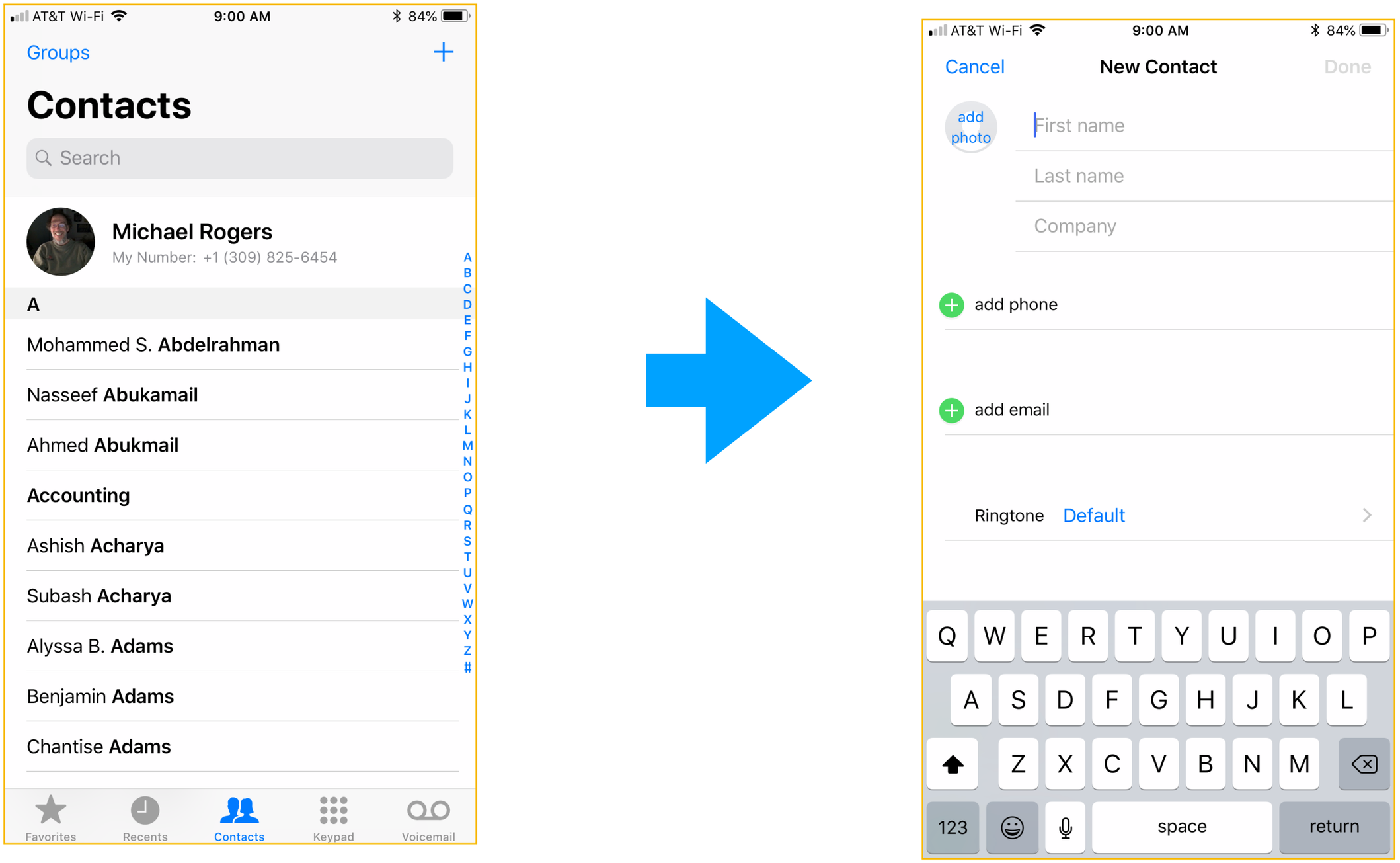
- Sometimes we need to interrupt the flow of control through an app, for instance:
    - present a screen so the user can provide new data, or edit existing data
    - notify/caution the user about something
    - present a user with a set of actions to choose from



- This is a **modal** experience: the user is in a particular mode, and cannot get out of it until they respond and make the presented view controller go away

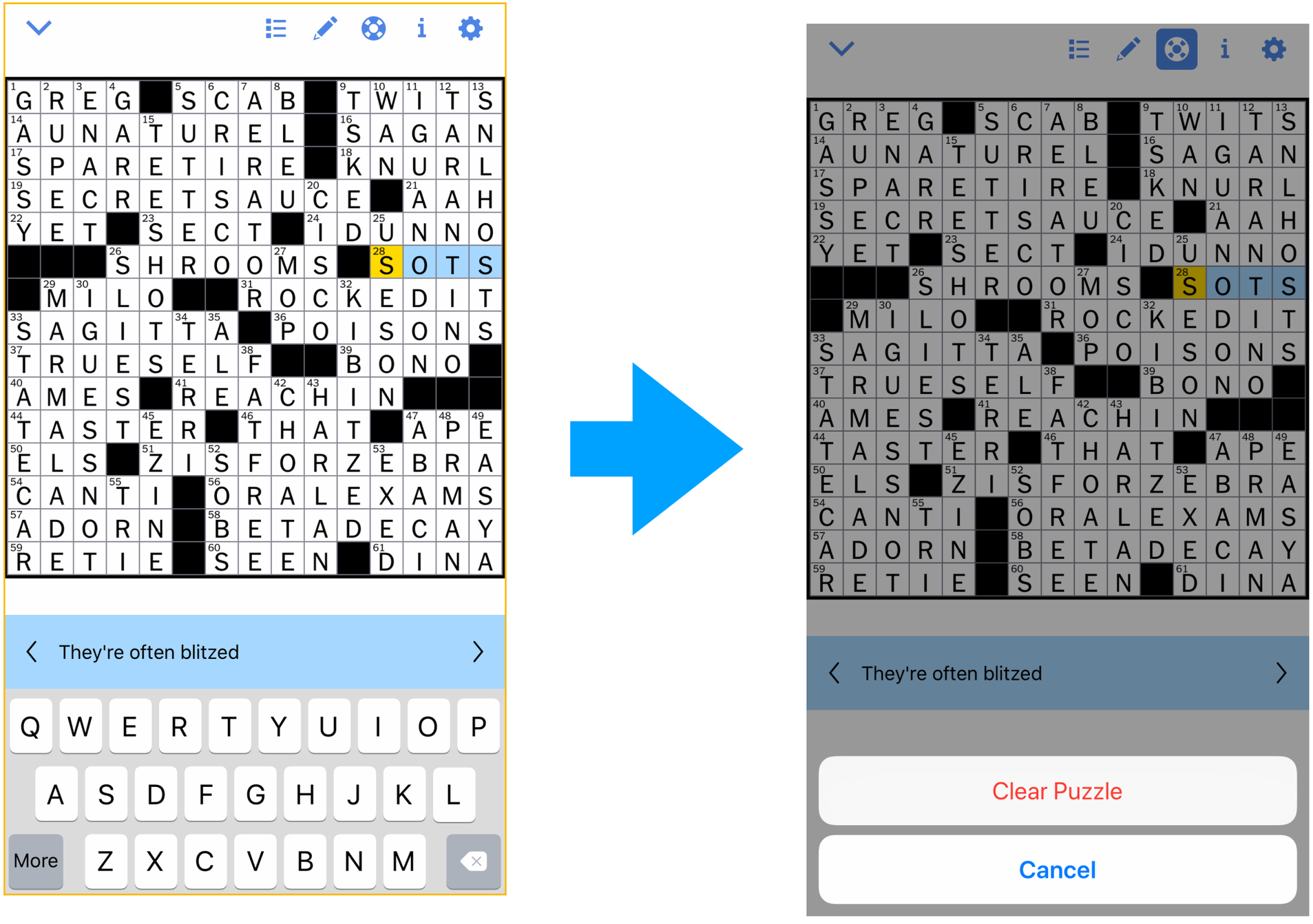
- A modal view is **presented**, rather than shown: it **totally obscures** any container view controller -- navigation controller, tab bar controller or split view controller -- that happens to be in effect when it is presented, and is not affected by it: it is not pushed onto the stack, added to a TBVC's viewControllers array, etc.

Modal Experience: Gathering Data



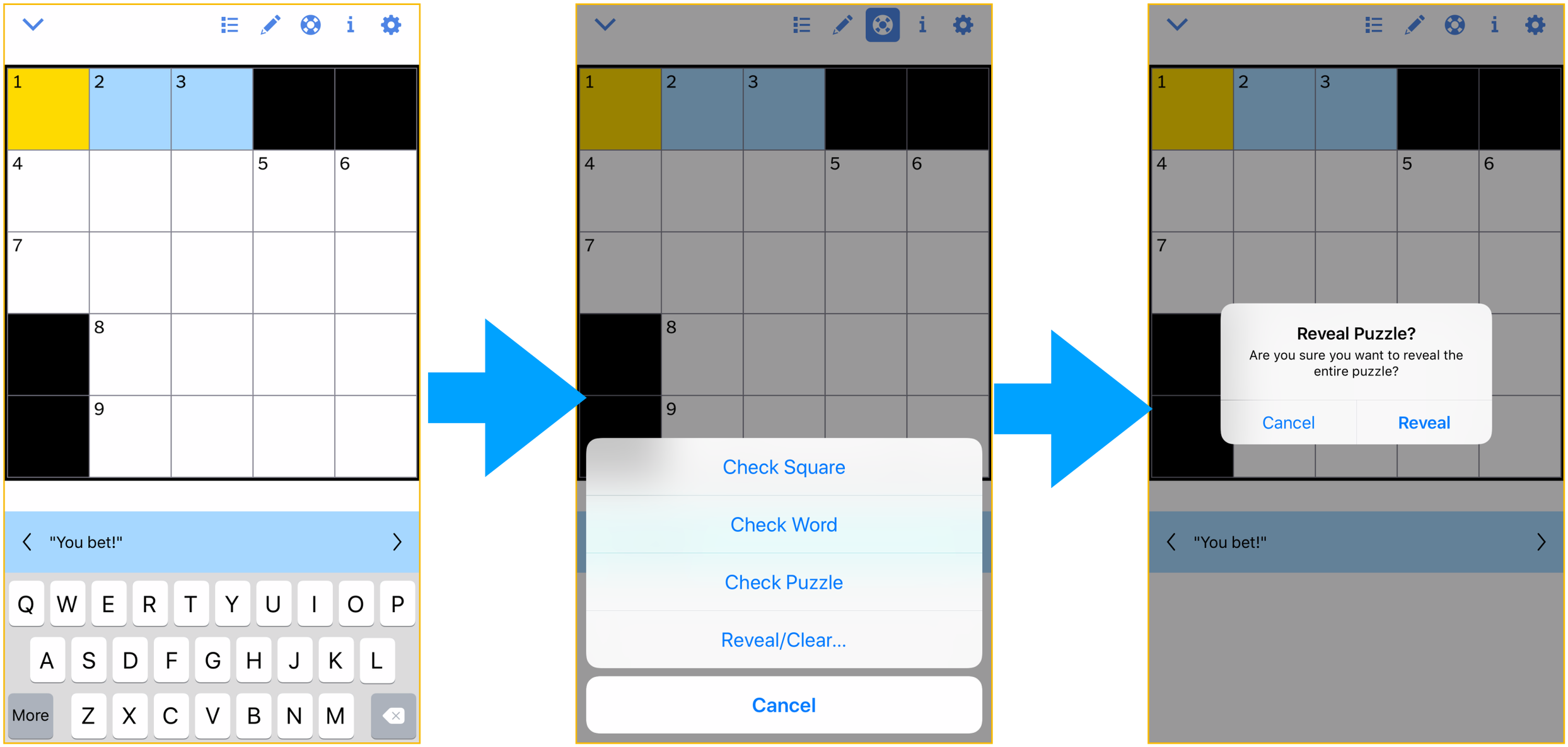
Contacts is a navigation-based app (in a tabbed app!), so we can navigate through our data (contacts) in the usual fashion (e.g., tapping on Ashish would get to more information about him): but tapping on + provides a different, modal experience

Modal Experience: Action Sheet



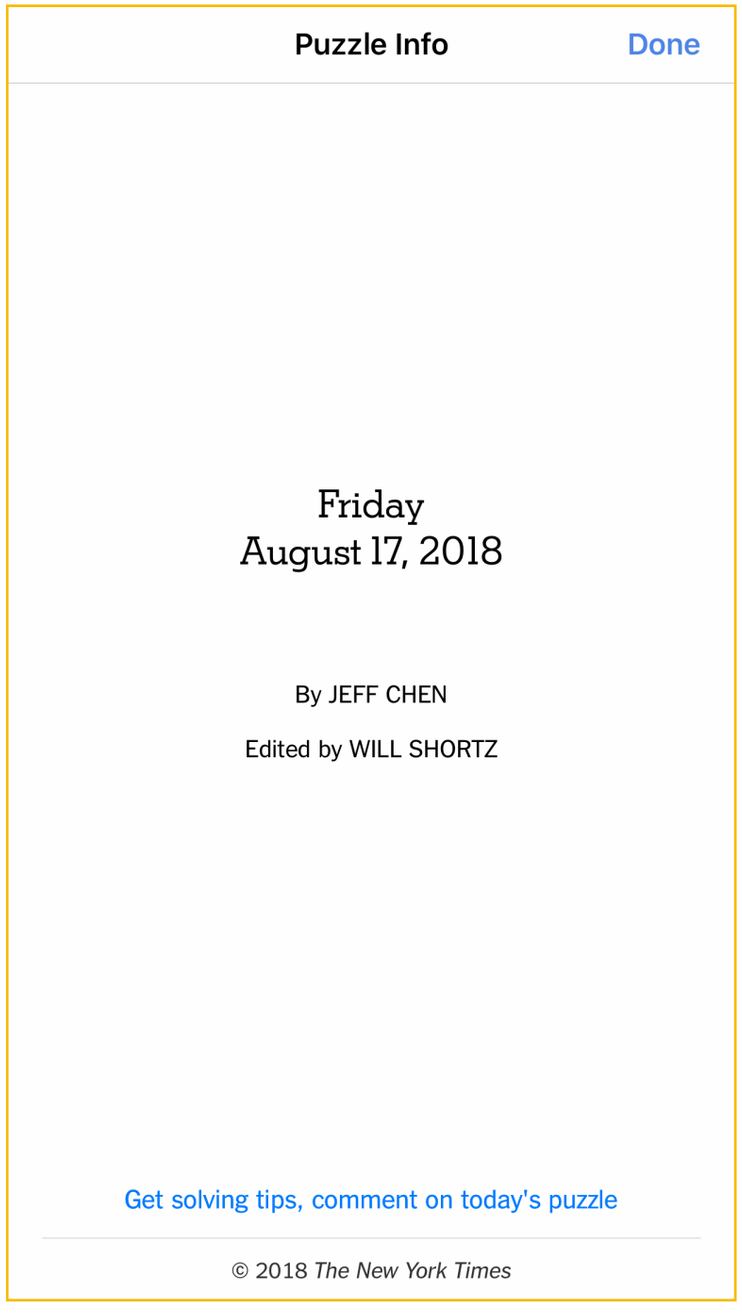
Once the user has tapped on the life-saver button, they are in a mode where they must respond to an action sheet before returning to the app

Modal Experience: Action Sheet and Alert



Here, the action sheet leads to an alert, another modal experience

Modal Experience:



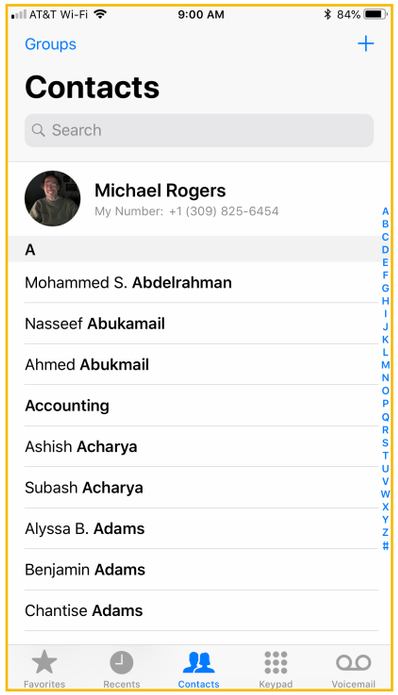
Terminology: Presenting View Controllers

- When we present a VC on top of another, the one that *was* visible is the **presenting view controller**; and the one that *becomes* visible is the **presented view controller**.

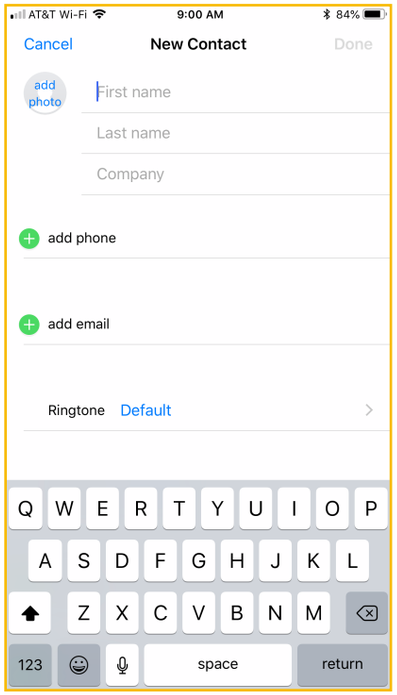
The *presenting* view controller

The *presented* view controller

- Presenting can be done modally:
  - **gather** more content (e.g., from a table view, click on a + button to add a user to a table view, and then fill in user details on the presented view controller)
  - **alert** the user that something has happened that needs their attention



Tap on row to present details



Gathering More Content: A Modal Model

- A very common situation is to gather data by displaying a table in a navigation controller, and then modally presenting a screen where new data can be added, or existing data edited.

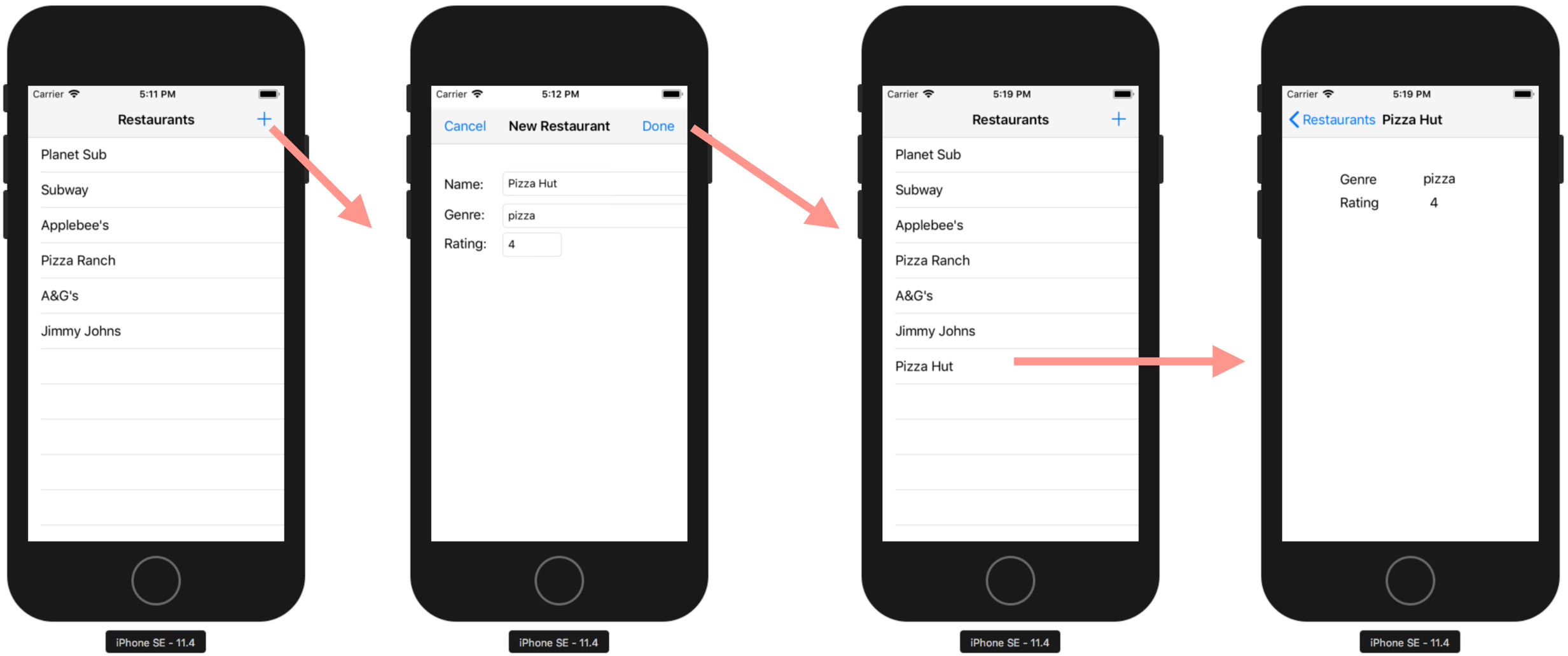
- We will illustrate this

  - in **Storyboard** by giving Best Restaurants the ability to add **new** restaurants

  - time permitting, in **code**, by looking again at NavigationControllers in Code - Airline Example (which already *has* this ability)

ICE: Gathering More Content (SB)

- When we are done, we will have achieved the following ...



ICE: Gathering More Content (SB)

1.Download and open up Best Restaurants-6.1-Completed

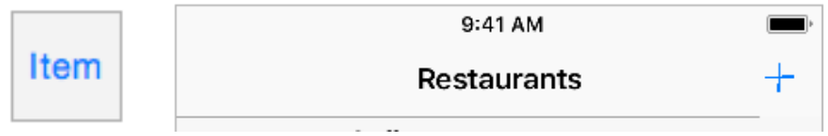
2.Create a UIViewController subclass, **NewRestaurantViewController** (NRVC), with a Restaurant property, **newRestaurant**, and outlets for 3 textfields — **nameTF**, **genreTF** and **ratingTF**.

```
class NewRestaurantViewController: UIViewController {
    var newRestaurant:Restaurant!

    @IBOutlet weak var nameTF:UITextField!
    @IBOutlet weak var genreTF:UITextField!
    @IBOutlet weak var ratingTF:UITextField!
```

3.In storyboard, drag in a UIViewController, and change its identity to NRVC.

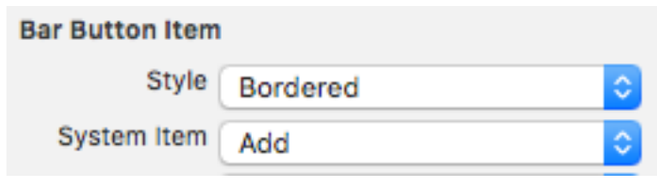
4.Place it beneath the RestaurantTableViewController

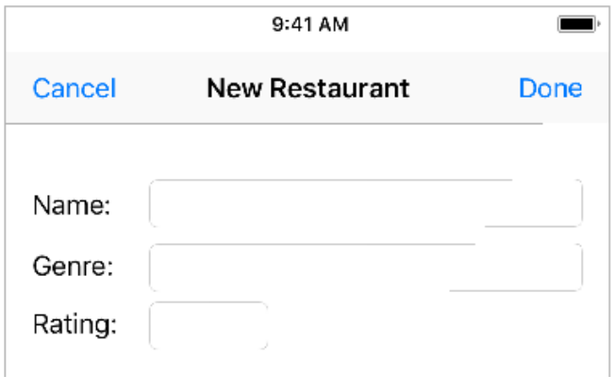
5.Drag a UIBarButtonItem into RestaurantTVC's navigation bar, on the right, and change its System Item attribute to Add



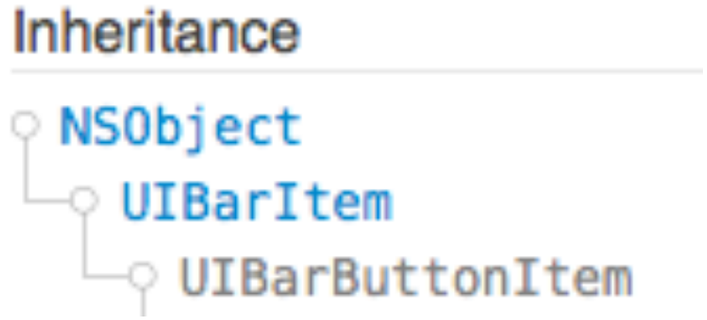
6.In NewRestaurant's view:

1. add a navigation bar at the top of the view, titled **New Restaurant**
2. add 2 UIBarButtonItems into that navigation bar, and change their System Item attributes to **Cancel** and **Done**, respectively
3. create actions **cancel(:_)** and **done(:_)** in NewRestaurantVC (control-drag from the bar button items to NewRestaurantVC)
4. add labeled text fields for name, genre, and rating, and connect them to the outlets in code





Techy Aside*: What's a UIBarButtonItem?



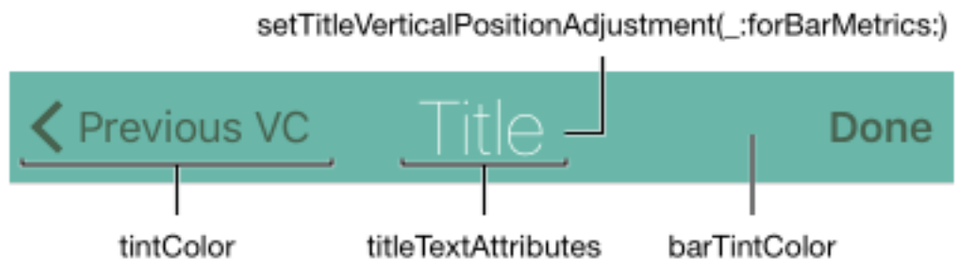
- A particular type of button — not a subclass of UIButton — designed to fit in a toolbar or navigation bar

- It has a target, action and style

```
enum UIBarButtonItemStyle : Int {
    case Plain
    case Bordered
    case Done
}
```

**as usual, techy asides may be omitted on a first reading*

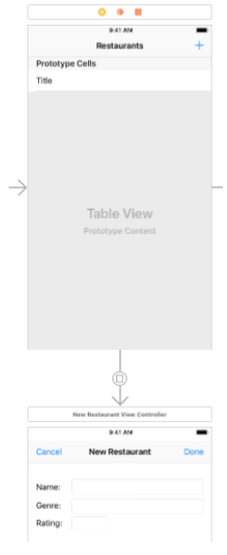
Techy Aside: Getting the Colors Right



- barTintColor = color of navigation bar

- tintColor = color of button images and titles

ICE: Gathering More Content (SB)



7. Create a segue from the + (Add) button in RestaurantsTVC to the NewRestaurantVC.

8. There are now 2 segues leading from RestaurantTVC, so we will need to modify its prepare(for segue:sender:) method so it *conditionally* executes:

```
override func prepare(for segue: UIStoryboardSegue, sender: Any?) {
 // only if we are segueing to the Restaurant Details screen, must we populate that VC
   if segue.identifier == "Restaurant Details" {
     let restaurantDVC = segue.destination as! RestaurantDetailsViewController
     // Pass the selected object to the new view controller.
     restaurantDVC.restaurant = Restaurants.shared[tableView.indexPathForSelectedRow!.row]
   }
}
```

ICE: Gathering More Content (SB)

9.Complete NewRestaurantVC's done(_:) and cancel(_:) actions

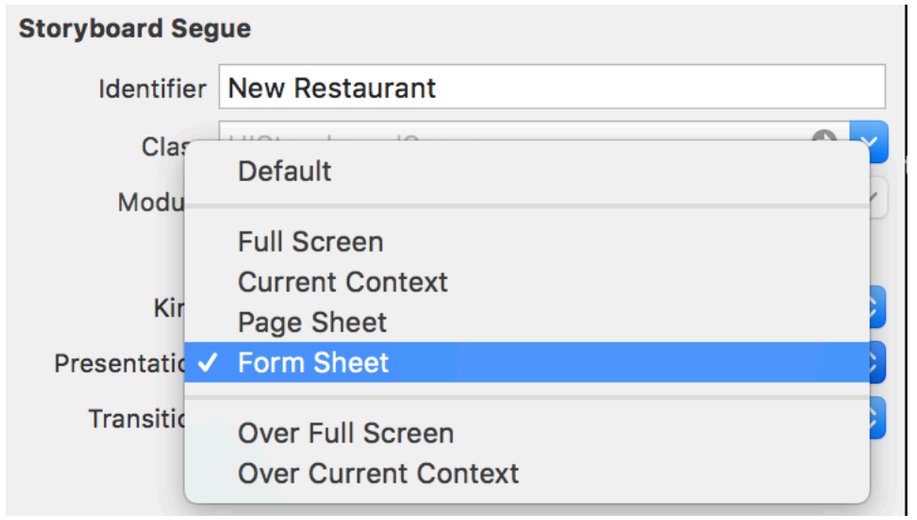
```
@IBAction func done(_ sender: Any) {
 let name = nameTF.text!
 let genre = genreTF.text!
 let rating = ratingTF.text!
 Restaurants.shared.add(restaurant:
                                    Restaurant(name: name, genre: Genre(rawValue:genre)!,
                                               rating: Int(rating)!))
  self.dismiss(animated: true, completion: nil)
}

@IBAction func cancel(_ sender: Any) {
 self.dismiss(animated: true, completion: nil)
}
```

10.Modify RestaurantTVC to force the table view to reload its data (otherwise, you won't see the new restaurant)

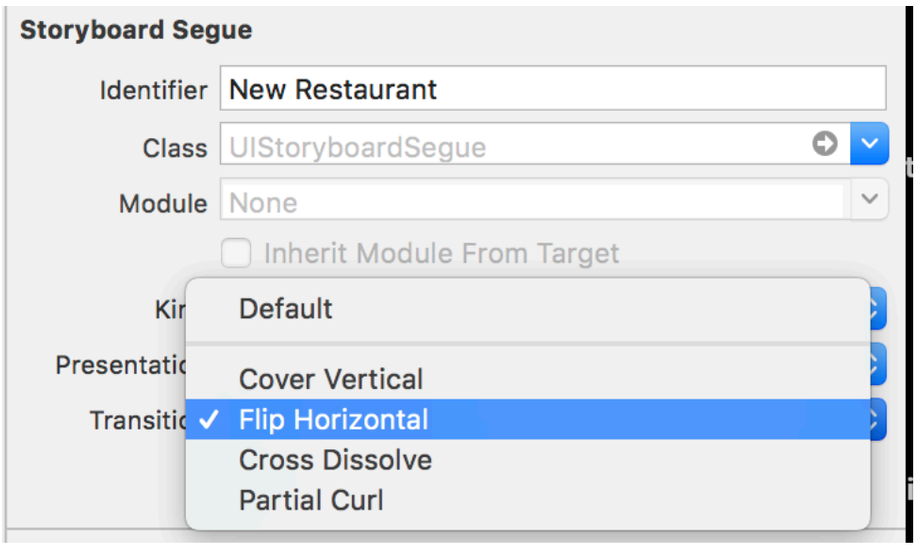
```
override func viewWillAppear(_ animated: Bool) {
  tableView.reloadData()
}
```

Presentation and Transition Styles

- The *appearance* of a presented view controller depends on its **presentation style** and the size/orientation of the device being used: it might take up the entire screen, or just a portion (showing the presenting view controller behind it, dimmed out)



- **Animation** is used during the presenting of a view controller: the details of the animation depend upon the **transition style**



- Presentations can be done in storyboard, using segues, or programmatically.

ICE: The Presenting View Controllers App

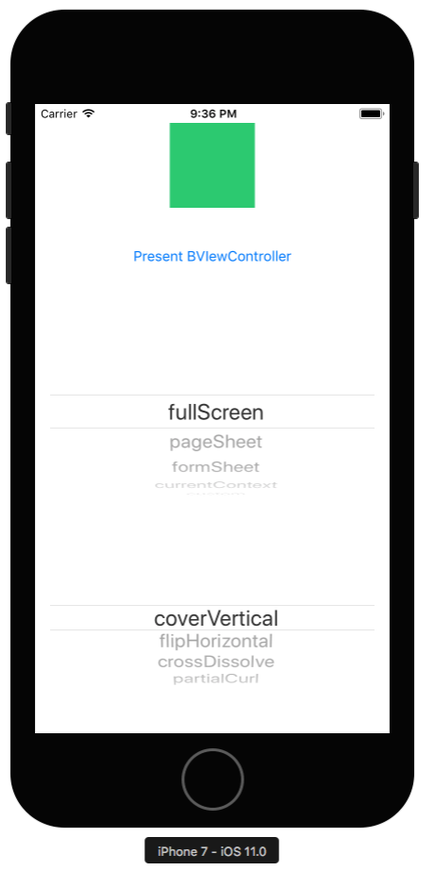
1. Download the Presenting View Controllers App

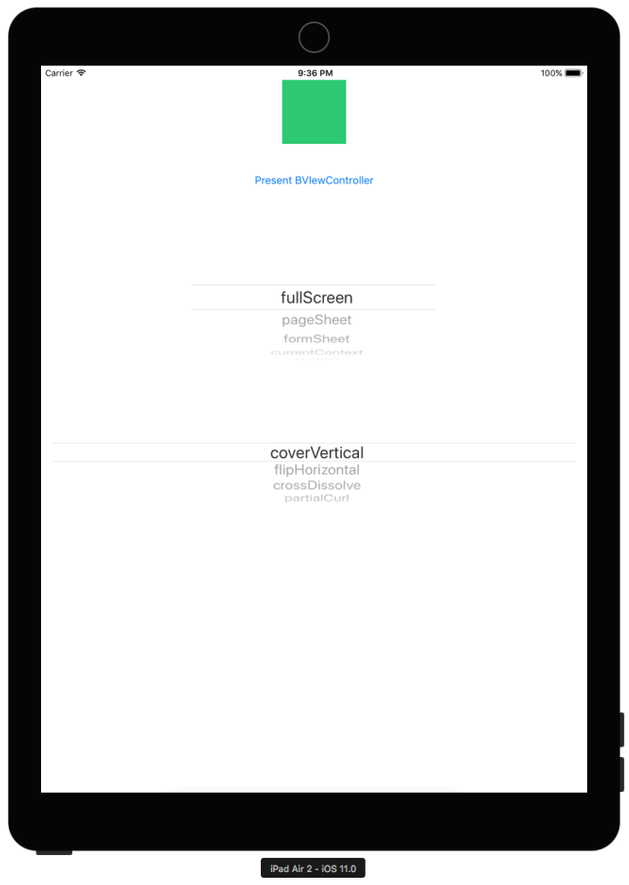
2. Launch it using an iPad Air 2

3. Try full screen, page sheet, form sheet and popup, in both landscape and portrait modes

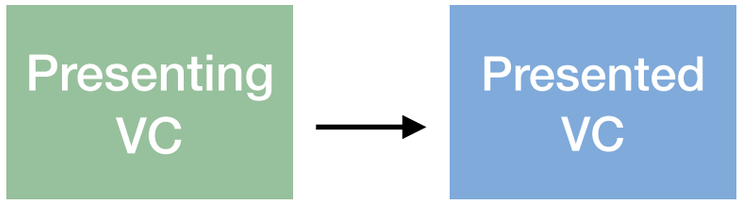
4. Repeat with an iPhone 6 or 7 (but not Plus)

5. Notice the difference in screen coverage





Presentation Styles



- The presentation styles are:

fullscreen pageSheet formSheet popOver

- The first 3 are full screen presentation styles: they take up most or all of the screen, and the view beneath is dimmed out. A popover just occupies a portion of the screen (depending on the device/ orientation)

- There is just one teensy-weensy complication: while this looks fine and dandy on an iPad, what about on an iPhone, where the width is much more compact?

So Many Screens, So Little Time

- When presenting a UIViewController, programmers used to have to just worry about the orientation (portrait or landscape), as there was only 1 iPhone, and 1 size screen.

- Now there are 3 iPhone screen sizes, 3 iPad screen sizes, and a View Controller might appear in a split screen, so the appearance of the view controller might need to be adapted depending upon the situation

- To do this, Apple has created horizontal and vertical **size classes** and each can be either **compact** or **regular**.

- How a presented view controller appears depends upon whether the horizontal class is compact or regular.

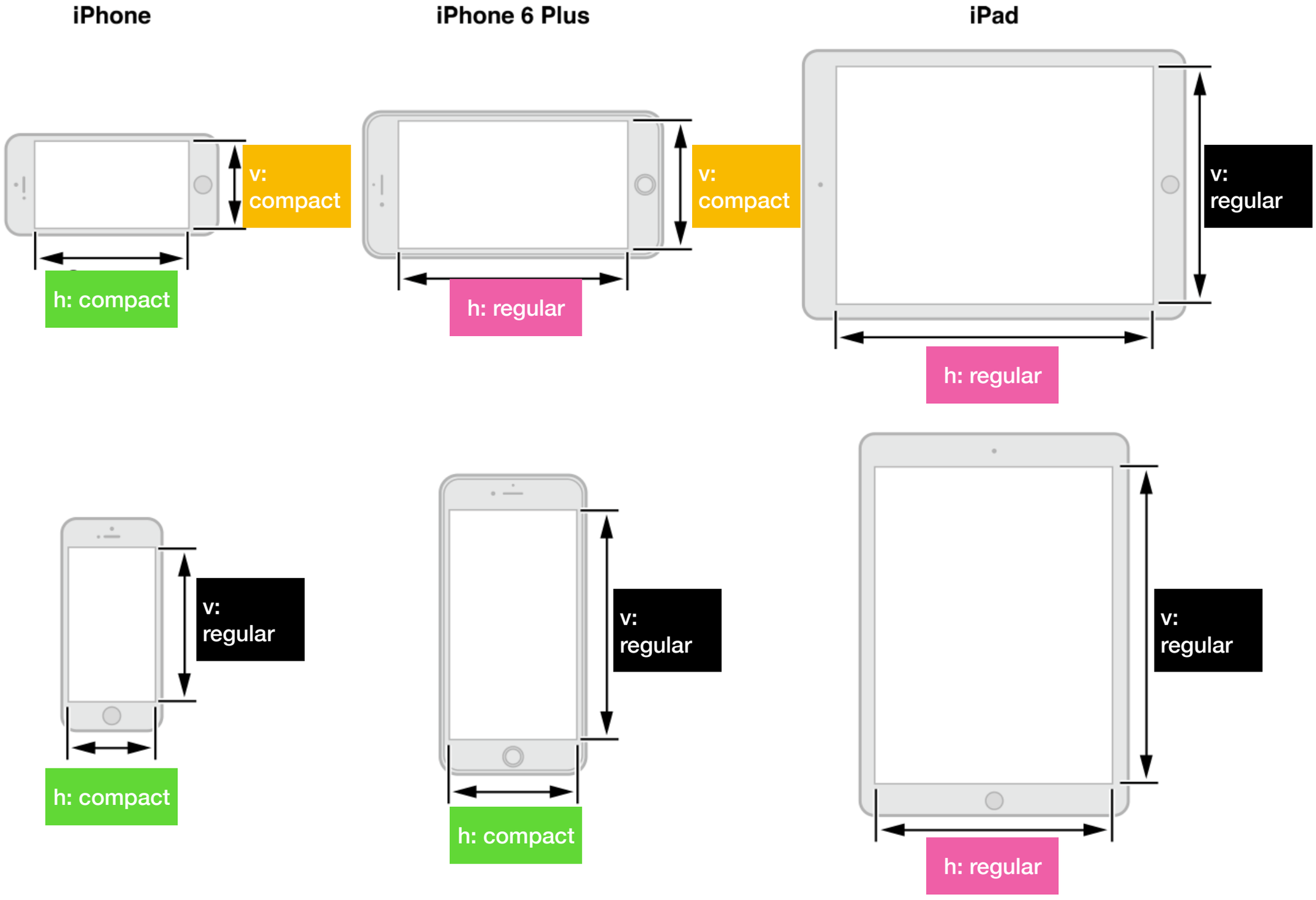
Size Classes

Horizontally Compact

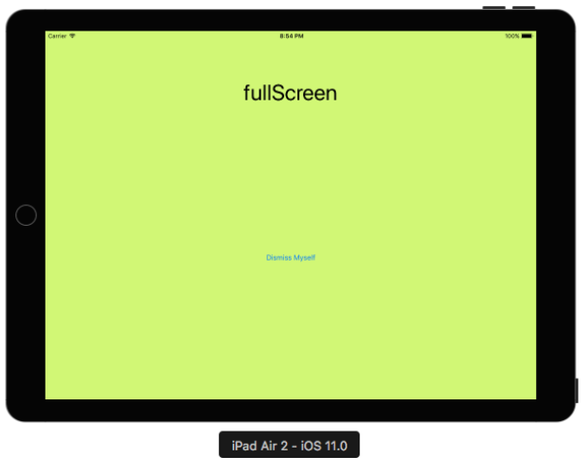
Vertically Compact

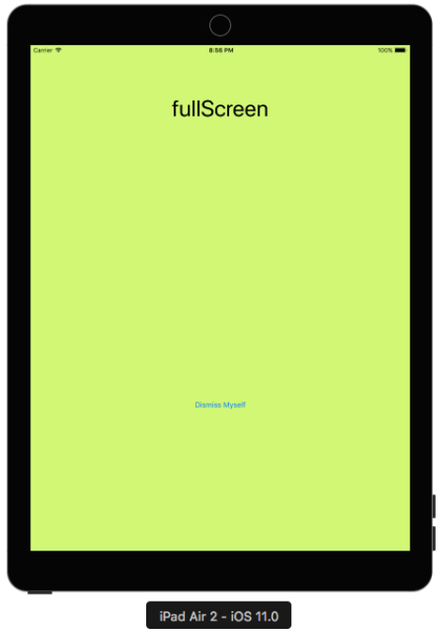
Horizontally Regular

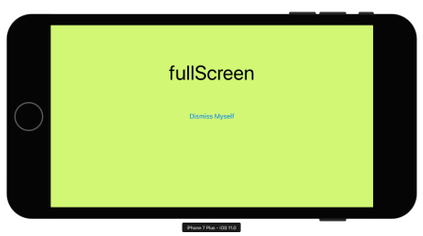
Vertically Regular



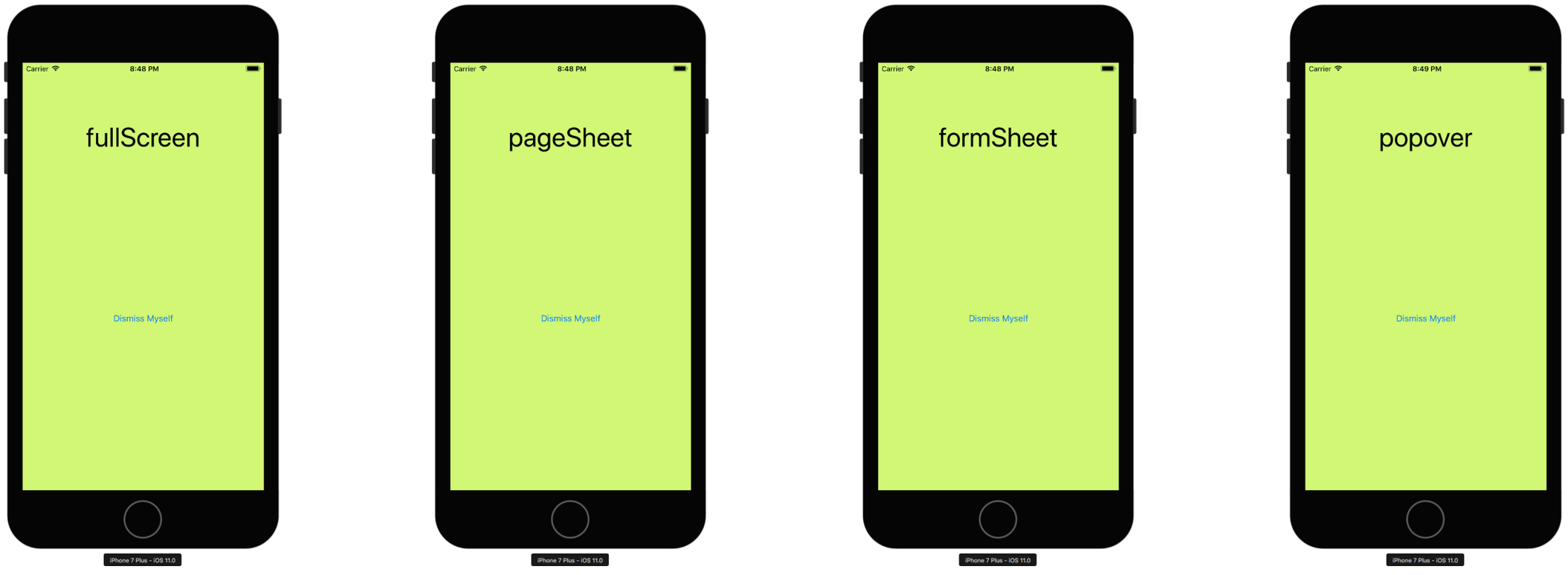
Presentation Styles with Horizontally Regular Size Classes(iPad, or iPhone 6/7Plus in landscape)







Presentation Styles with Horizontally Compact Size Classes (iPhone & iPhone 6/7 Plus in Portrait)



Key point: with a horizontally compact size class, <u>all</u> presented UIViewControllers take up the <u>entire</u> screen 🧐

Exercises

- Pending ...

References

- https://developer.apple.com/library/content/featuredarticles/ViewControllerPGforiPhoneOS/index.html#//apple_ref/doc/uid/TP40007457-CH2-SW1

- https://developer.apple.com/documentation/uikit/uitraitcollection