Stanford CS193p

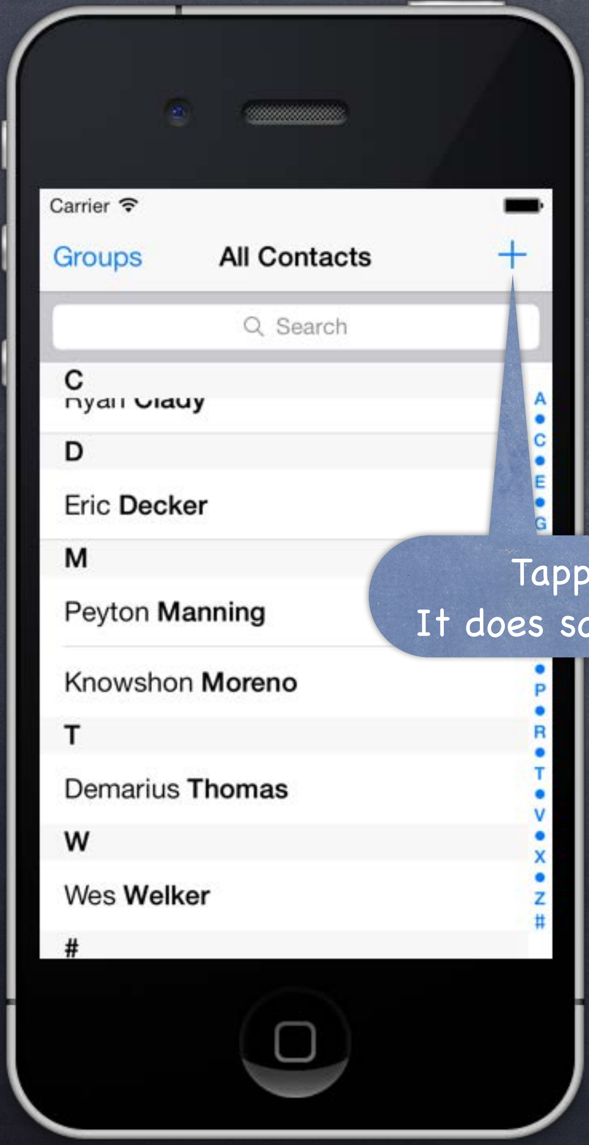
Developing Applications for iOS
Winter 2017

Today

- More Segues
  - Modal
  - Unwind
  - Popover
  - Embed

Modal View Controllers

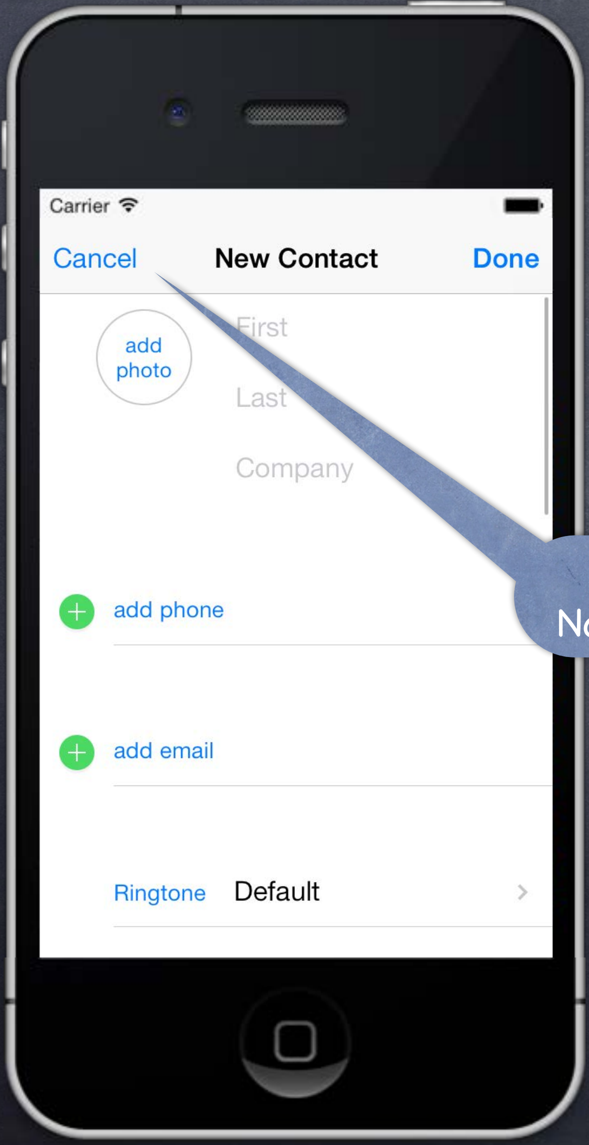
- A way of segueing that takes over the screen
  - Should be used with care.
- Example
  - Contacts application.



Tapping here adds a new contact.
It does so by taking over the entire screen.

Modal View Controllers

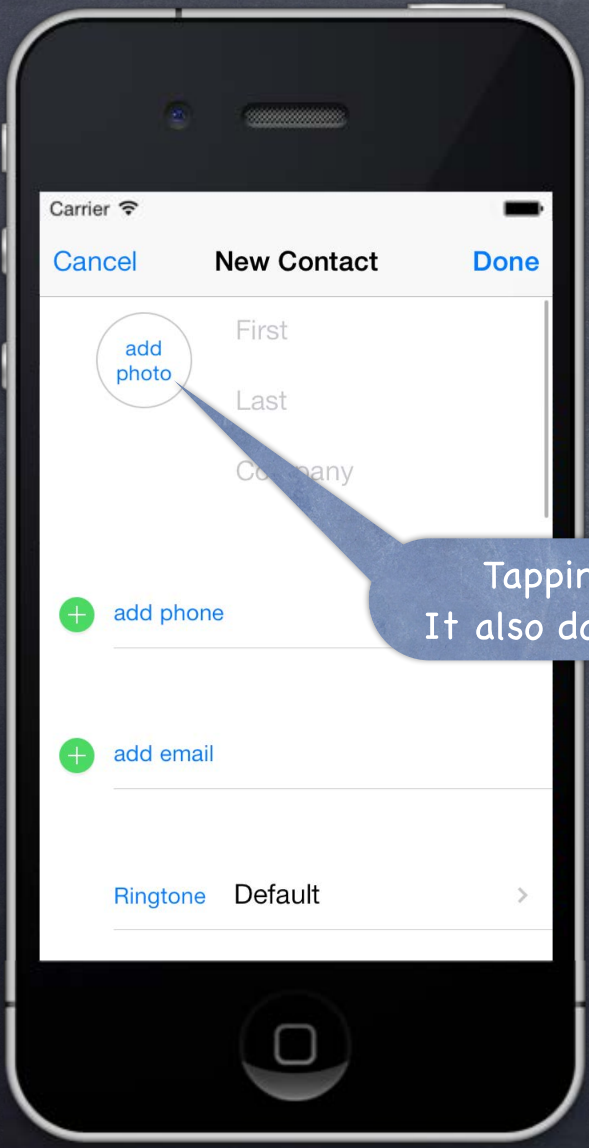
- A way of segueing that takes over the screen
  - Should be used with care.
- Example
  - Contacts application.



This is not a push.
Notice, no back button (only Cancel).

Modal View Controllers

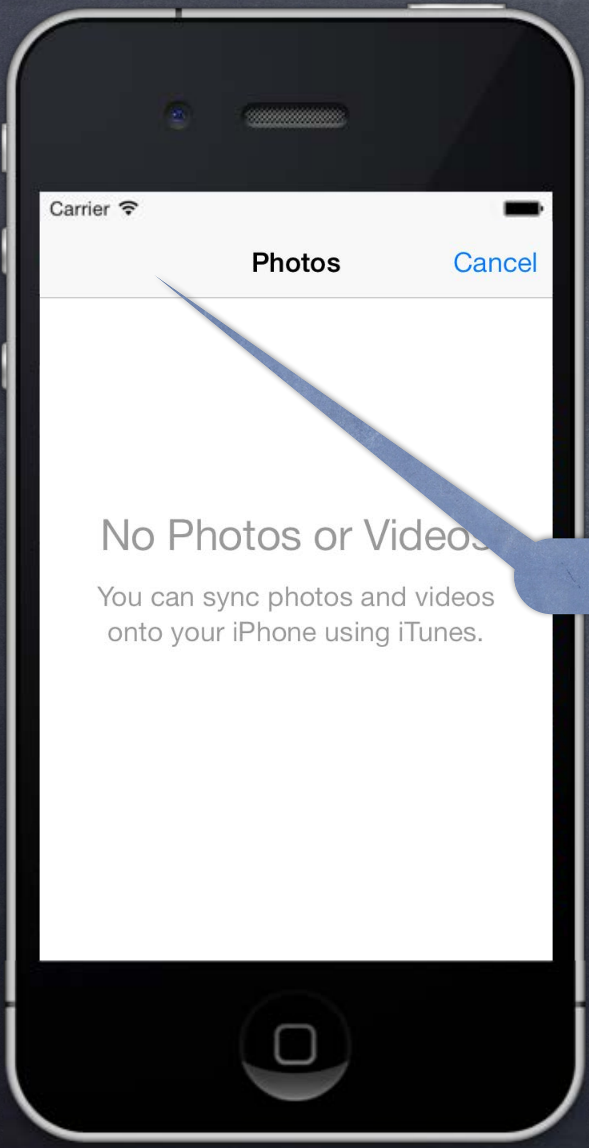
- A way of segueing that takes over the screen
  - Should be used with care.
- Example
  - Contacts application.



Tapping here adds a photo to this contact.
It also does so by taking over the entire screen.

Modal View Controllers

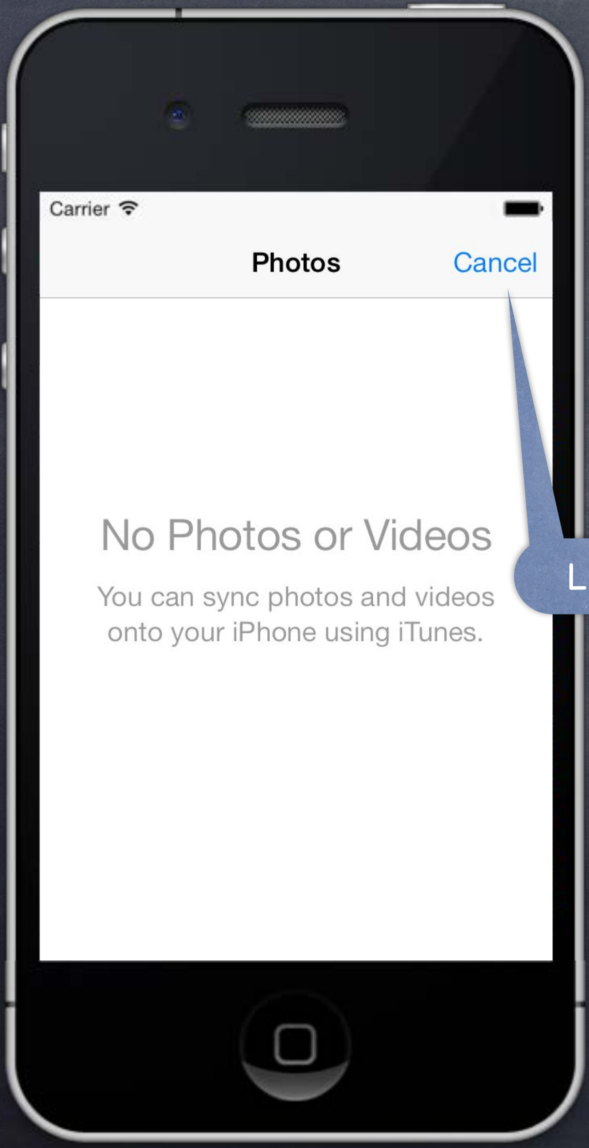
- A way of segueing that takes over the screen
  - Should be used with care.
- Example
  - Contacts application.


Carrier
Photos
Cancel
No Photos or Videos
You can sync photos and videos onto your iPhone using iTunes.

Again, no back button.

Modal View Controllers

- A way of segueing that takes over the screen
  - Should be used with care.
- Example
  - Contacts application.



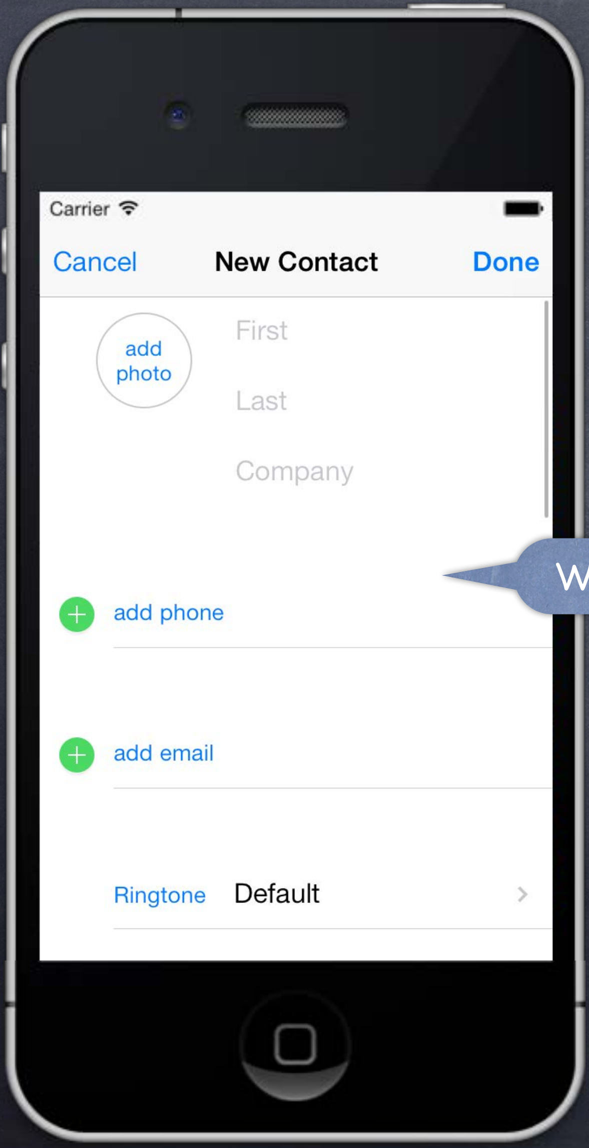
No Photos or Videos

You can sync photos and videos onto your iPhone using iTunes.

Let’s Cancel and see what happens.

Modal View Controllers

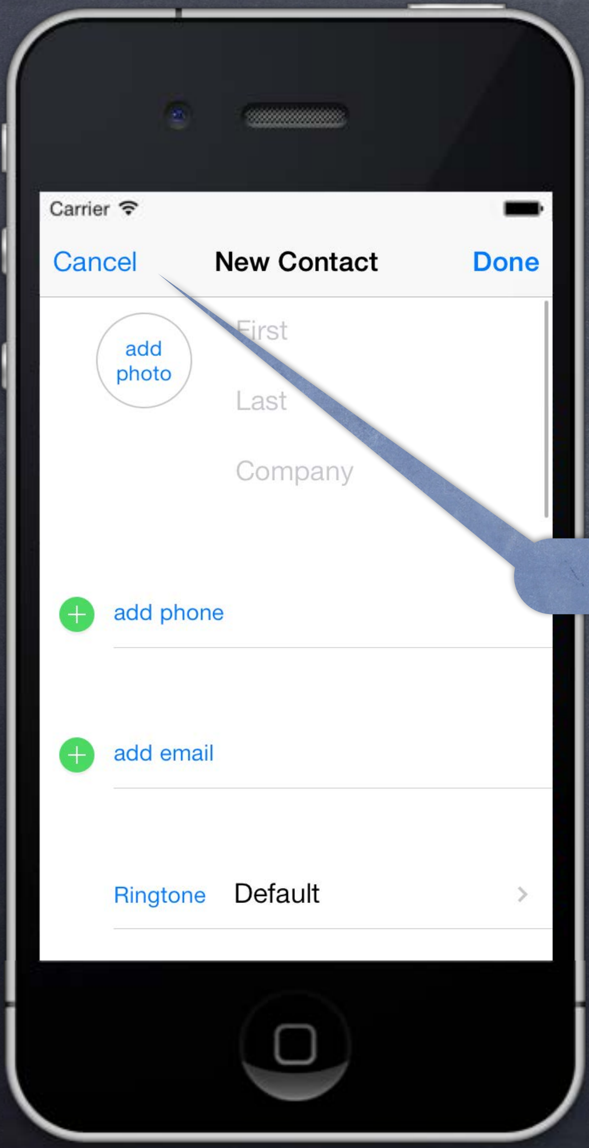
- A way of segueing that takes over the screen
  Should be used with care.
- Example
  Contacts application.



We're back to the last Modal View Controller.

Modal View Controllers

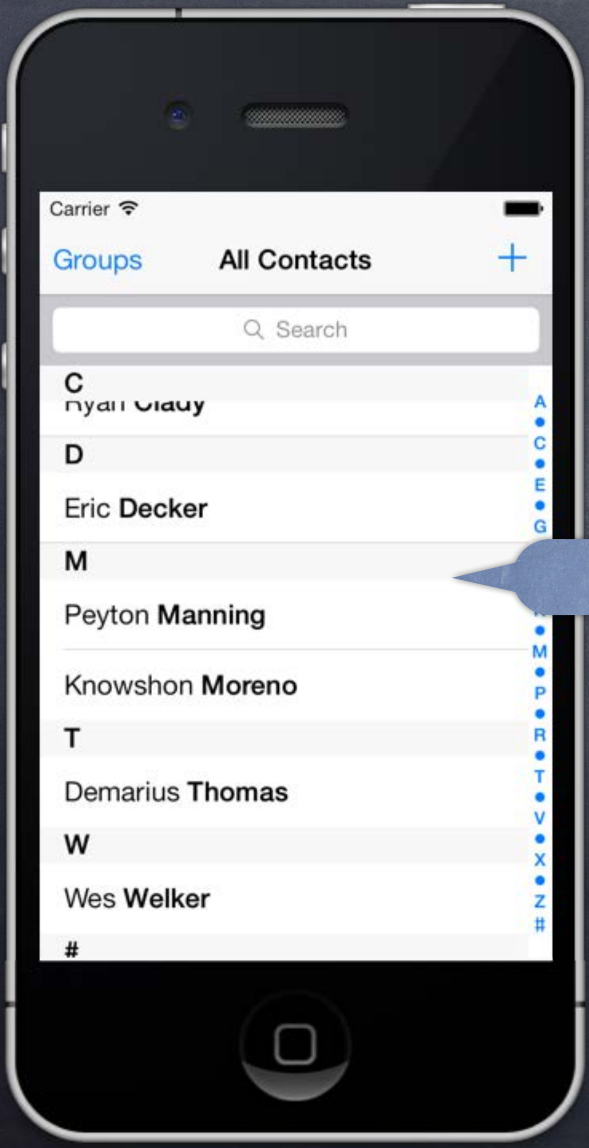
- A way of segueing that takes over the screen
  - Should be used with care.
- Example
  - Contacts application.



And Cancel again …

Modal View Controllers

- A way of segueing that takes over the screen
  - Should be used with care.
- Example
  - Contacts application.



Back to where we started.

Modal View Controllers

- Considerations

  The view controller we segue to using a Modal segue will take over the entire screen

  This can be rather disconcerting to the user, so use this carefully

- How do we set a Modal segue up?

  Just ctrl-drag from a button to another View Controller & pick segue type "Modal"

  Inspect the segue to set the style of presentation

  If you need to present a Modal VC not from a button, use a manual segue …

  `func performSegue(withIdentifier: String, sender: Any?)`

  … or, if you have the view controller itself (e.g. Alerts or from `instantiateViewController`) …

  `func present(UIViewController, animated: Bool, completion: (() -> Void)? = nil)`

  In horizontally regular environments, `modalPresentationStyle` will determine how it appears …

  `.fullScreen`, `.overFullScreen` (presenter left underneath), `.popover`, `.formSheet`, etc.

  In horizontally compact environments, this will adapt to always be full screen!

Modal View Controllers

- Preparing for a Modal segue

  You prepare for a Modal segue just like any other segue …

```
func prepare(for: UIStoryboardSegue, sender: Any?) {
    if segue.identifier == “GoToMyModalVC” {
        let vc = segue.destination as MyModalVC
        // set up the vc to run here
    }
}
```

- Hearing back from a Modally segued-to View Controller

  When the Modal View Controller is “done”, how does it communicate results back to presenter?
  If there’s nothing to be said, just dismiss the segued-to MVC (next slide).
  To communicate results, generally you would Unwind (more on that in a moment).

Modal View Controllers

- How to dismiss a view controller

  The presenting view controller is responsible for dismissing (not the presented).
  You do this by sending the presenting view controller this message …

  ```
  func dismiss(animated: Bool, completion: (() -> Void)? = nil)
  ```

  … which will dismiss whatever MVC it has presented (if any).

  If you send this to a presented view controller, for convenience, it will forward to its presenter (unless it itself has presented an MVC, in which case it will dismiss that MVC).
  But to reduce confusion in your code, only send `dismiss` to the presenting controller.

  Unwind Segues (coming up soon) automatically dismiss (you needn't call the above method).

Modal View Controllers

- How is the modal view controller animated onto the screen?

  Depends on this property in the view controller that is being presented …

  ```
  var modalTransitionStyle: UIModalTransitionStyle
  .coverVertical   // slides the presented modal VC up from bottom of screen (the default)
  .flipHorizontal  // flips the presenting view controller over to show the presented modal VC
  .crossDissolve   // presenting VC fades out as the presented VC fades in
  .partialCurl     // only if presenting VC is full screen (& no more modal presentations coming)
  ```

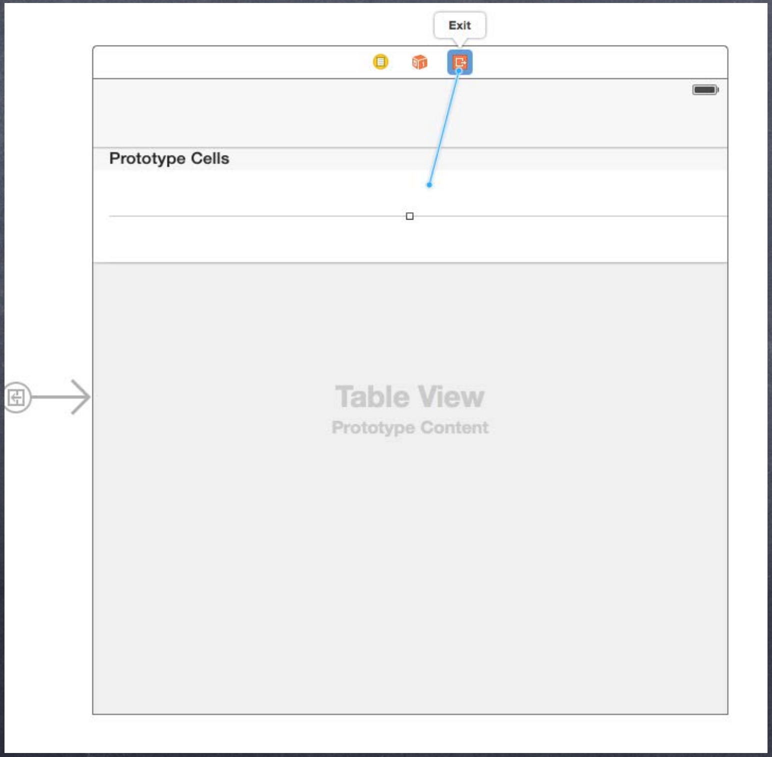
  You can also set this in the storyboard by inspecting the modal segue.

Unwind Segue

- The only segue that does NOT create a new MVC
  - It can only segue to other MVCs that (directly or indirectly) presented the current MVC
- What's it good for?
  - Jumping up the stack of cards in a navigation controller (other cards are considered presenters)
  - Dismissing a Modally segued-to MVC while reporting information back to the presenter

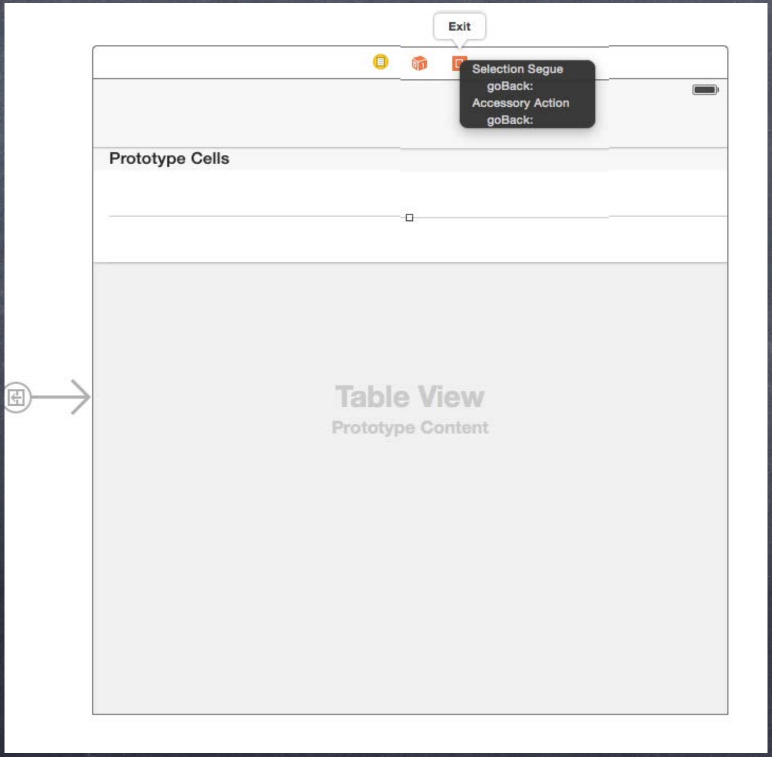
Unwind Segue

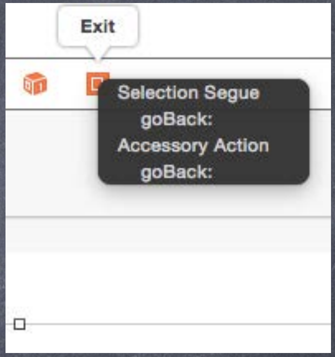
- How does it work?
  Instead of ctrl-dragging to another MVC, you ctrl-drag to the “Exit” button in the same MVC



Unwind Segue

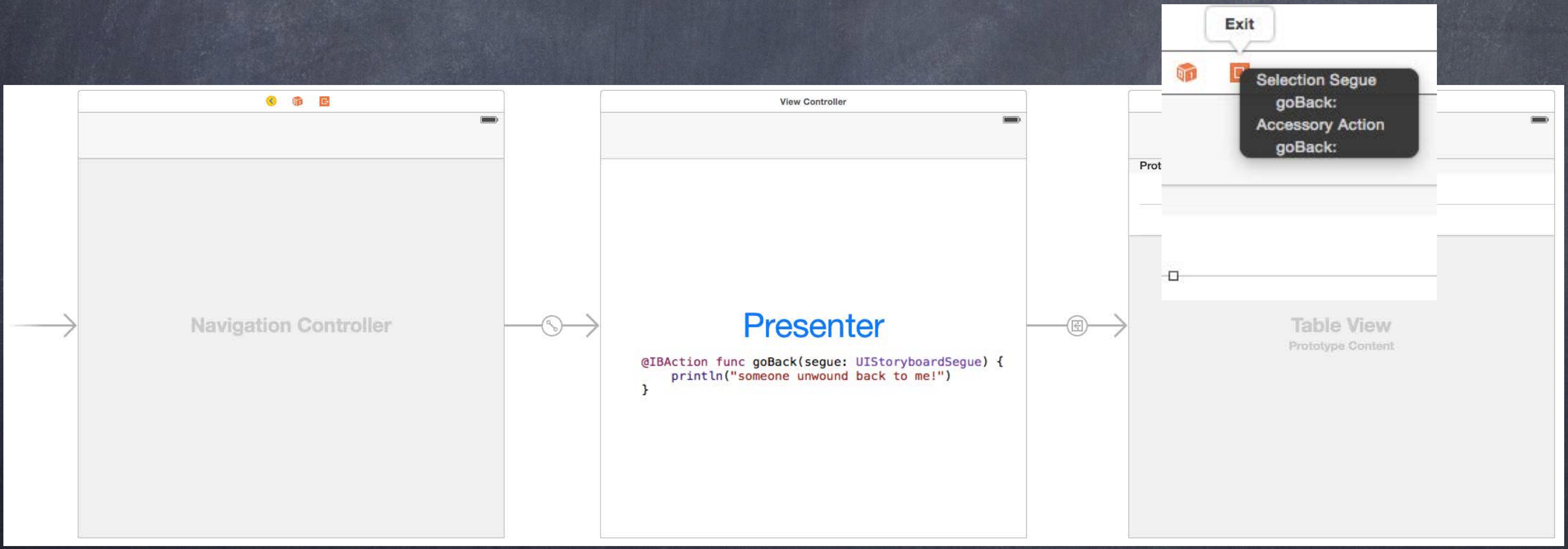
- How does it work?
  - Instead of ctrl-dragging to another MVC, you ctrl-drag to the "Exit" button in the same MVC





Unwind Segue

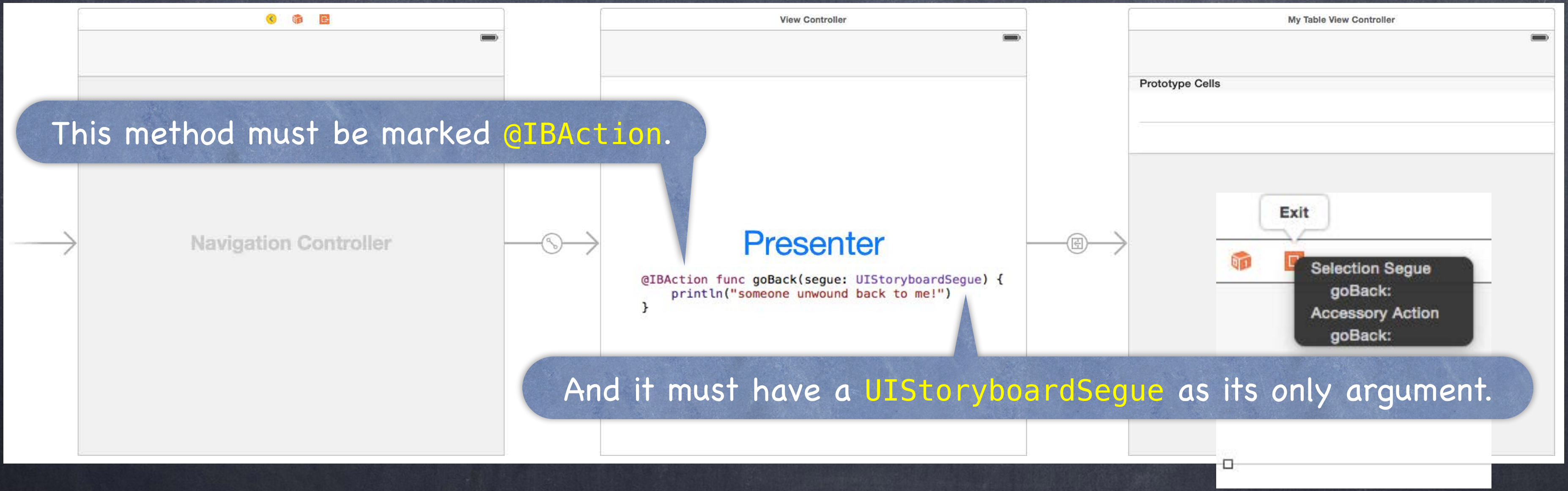
- How does it work?
  Instead of ctrl-dragging to another MVC, you ctrl-drag to the "Exit" button in the same MVC
  Then you can choose a special `@IBAction` method you've created in another MVC



```
@IBAction func goBack(segue: UIStoryboardSegue) {
    println("someone unwound back to me!")
}
```

Unwind Segue

- How does it work?
  - Instead of ctrl-dragging to another MVC, you ctrl-drag to the "Exit" button in the same MVC
  - Then you can choose a special @IBAction method you've created in another MVC
  - This means "segue by exiting me and finding a presenter who implements that method"
  - If no presenter (directly or indirectly) implements that method, the segue will not happen



```
@IBAction func goBack(segue: UIStoryboardSegue) {
    println("someone unwound back to me!")
}
```

This method must be marked @IBAction.

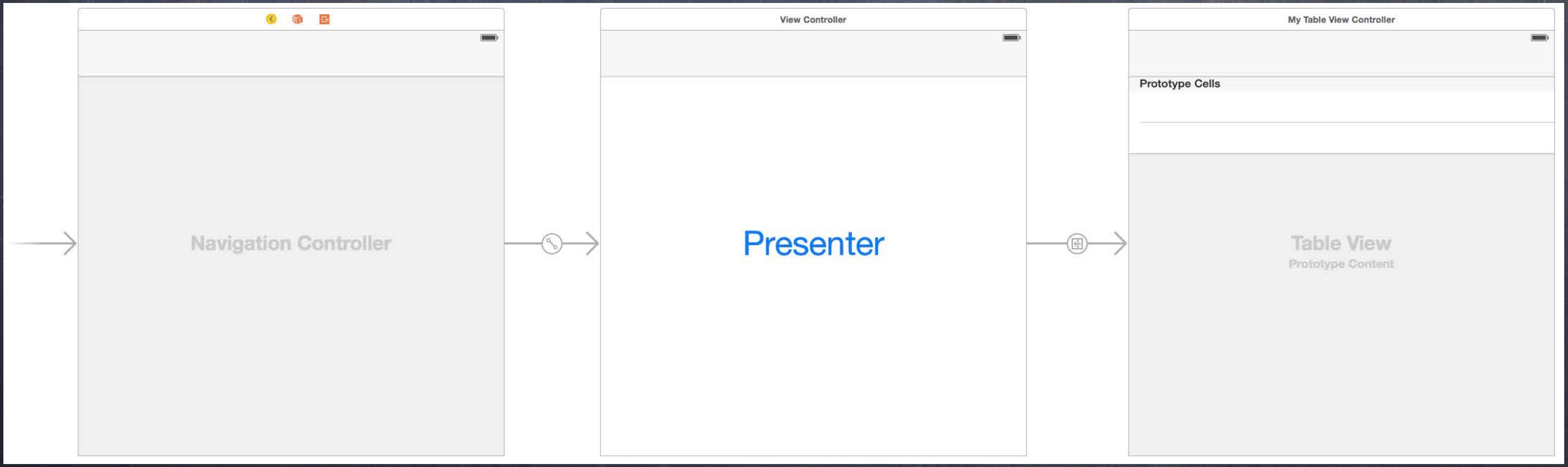
And it must have a UIStoryboardSegue as its only argument.

Unwind Segue

- How does it work?

    If the @IBAction can be found, you (i.e. the presented MVC) will get to prepare as normal

```
override func prepare(for segue: UIStoryboardSegue, sender: Any?) {
    if segue.identifier == "GoBack Unwind Segue" {
        unwoundToMVC = segue.destination as? MyPresentingMVC
        unwoundToMVC?.publicAPI = infoToCommunicateBack
    }
}
```



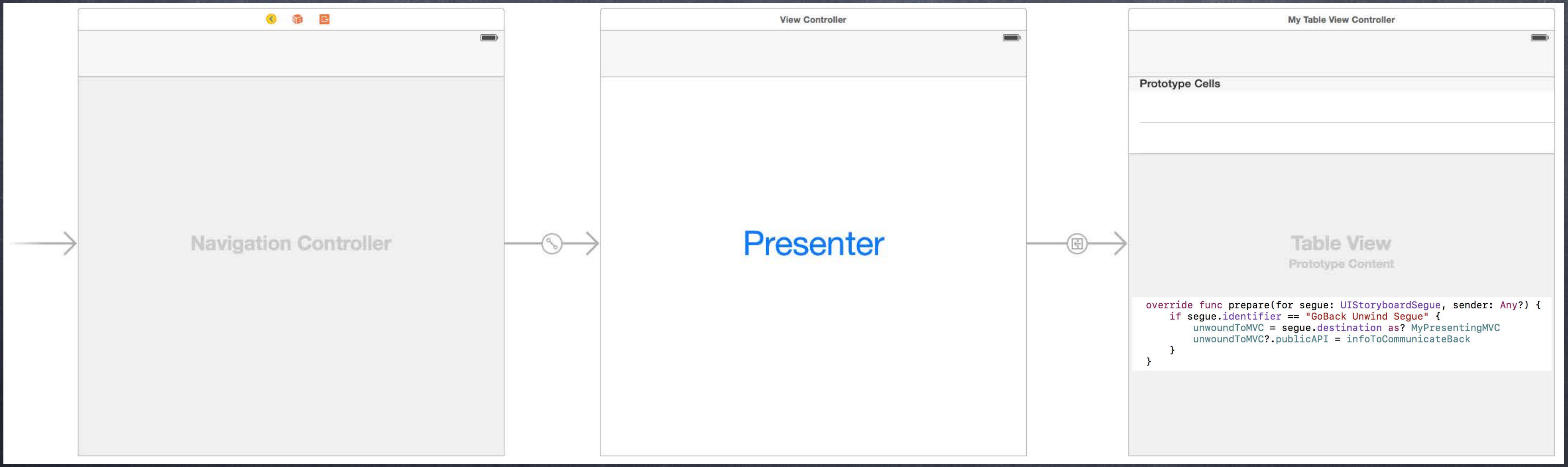
Unwind Segue

- How does it work?

  If the @IBAction can be found, you (i.e. the presented MVC) will get to prepare as normal

  Then the special @IBAction will be called in the other MVC and that MVC will be shown on screen

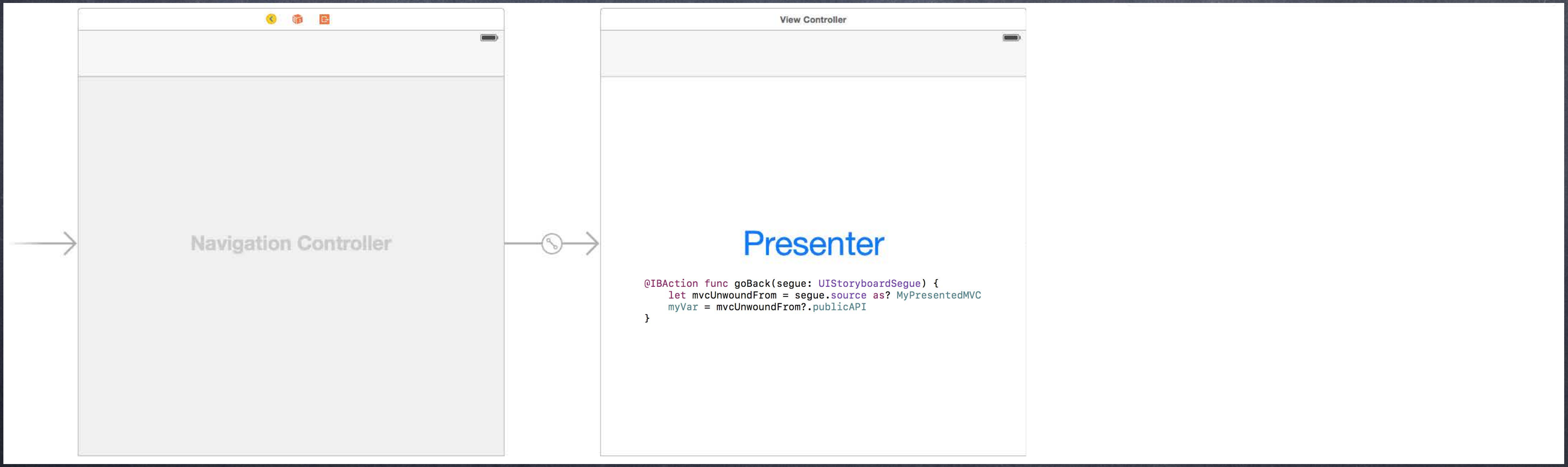
```
@IBAction func goBack(segue: UIStoryboardSegue) {
    let mvcUnwoundFrom = segue.source as? MyPresentedMVC
    myVar = mvcUnwoundFrom?.publicAPI
}
```



```
override func prepare(for segue: UIStoryboardSegue, sender: Any?) {
    if segue.identifier == "GoBack Unwind Segue" {
        unwoundToMVC = segue.destination as? MyPresentingMVC
        unwoundToMVC?.publicAPI = infoToCommunicateBack
    }
}
```

Unwind Segue

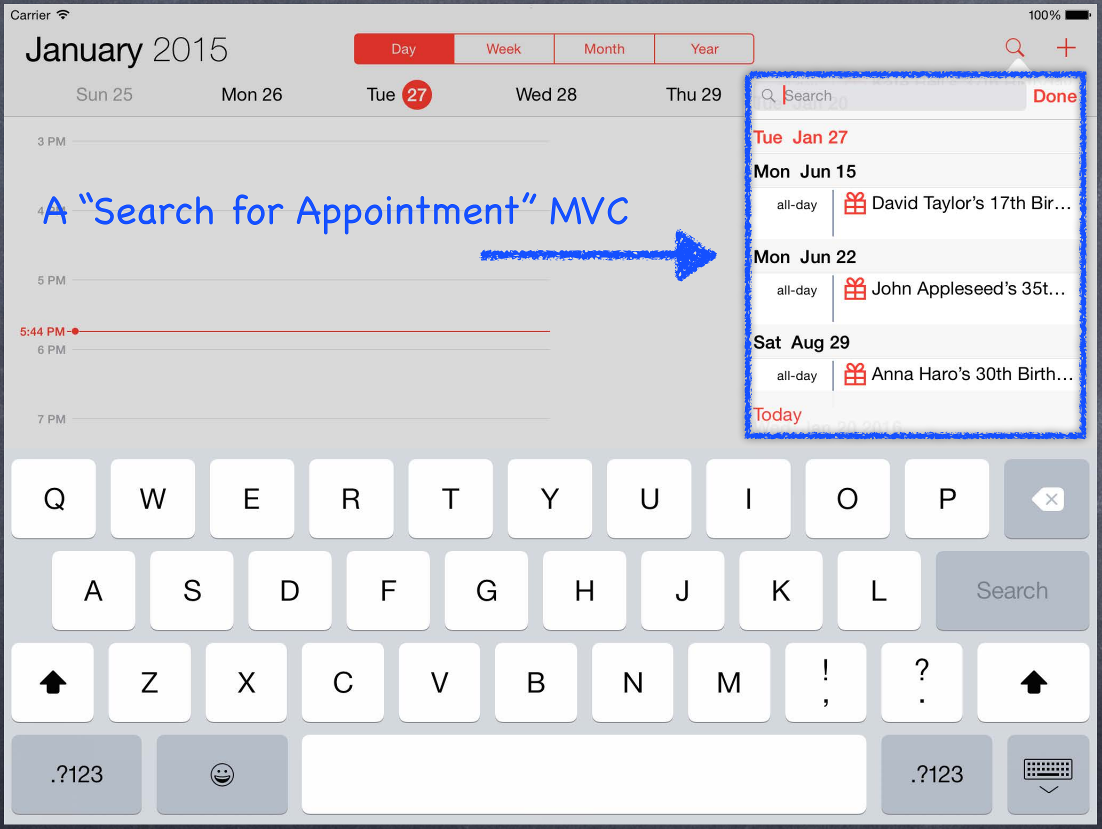
- How does it work?
  If the @IBAction can be found, you (i.e. the presented MVC) will get to **prepare** as normal
  Then the special @IBAction will be called in the other MVC and that MVC will be shown on screen
  You will be dismissed in the process (i.e. you'll be "unpresented" and thrown away)



```
@IBAction func goBack(segue: UIStoryboardSegue) {
    let mvcUnwoundFrom = segue.source as? MyPresentedMVC
    myVar = mvcUnwoundFrom?.publicAPI
}
```

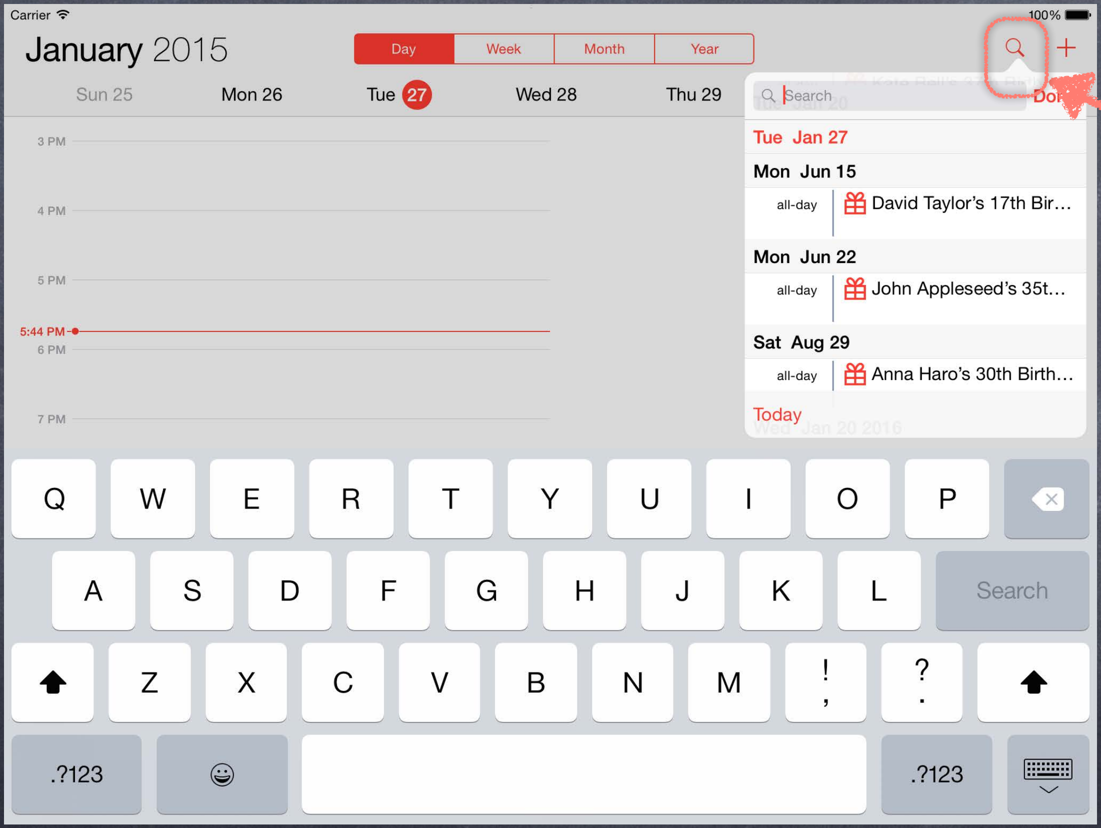
Popover

- Popovers pop an entire MVC over the rest of the screen



Popover

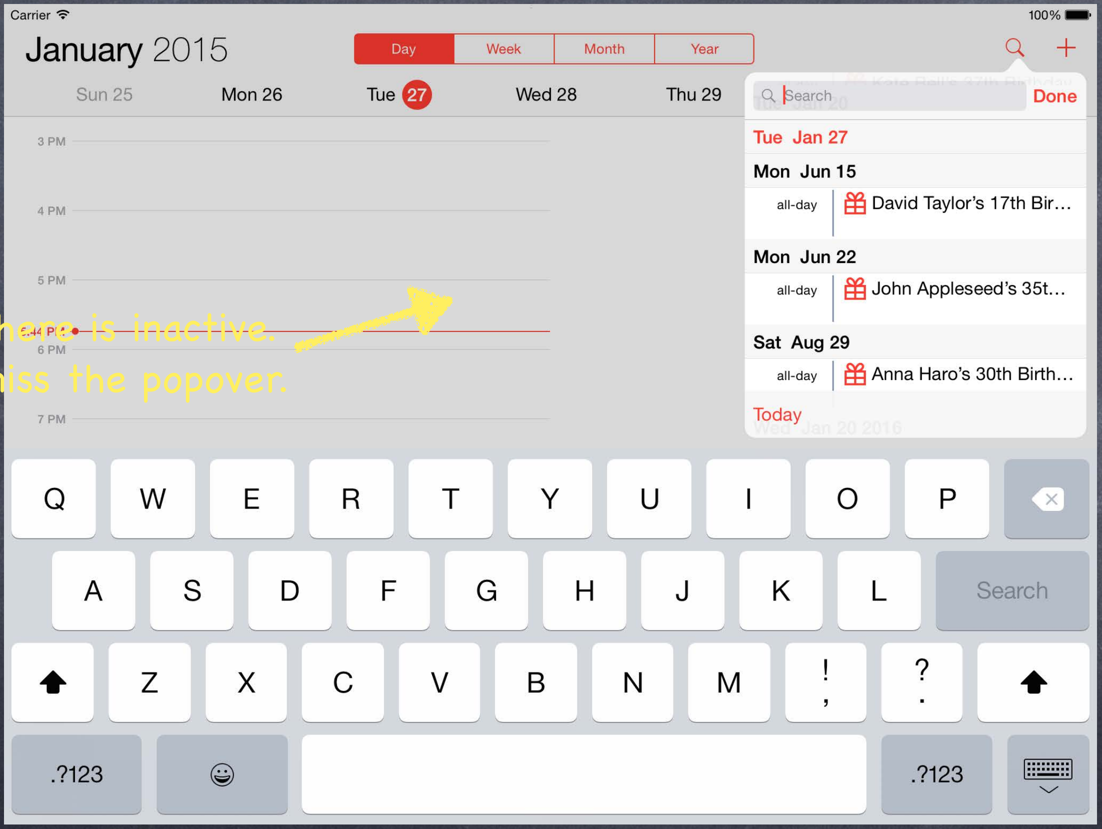
- Popovers pop an entire MVC over the rest of the screen



Popover's arrow pointing to what caused it to appear

Popover

- Popovers pop an entire MVC over the rest of the screen



The grayed out area here is inactive.
Touching in it will dismiss the popover.

CS193p
Winter 2017

Popover

- A popover is almost exactly the same as a Modal segue
  You still ctrl-drag, you still have an identifier, you still get to prepare
  It just looks a little different (but it's still "modal" in that you can't do anything else)
- Things to note when preparing for a popover segue
  All segues are managed via a `UIPresentationController` (but we're not going to cover that)
  But we are going to talk about a popover's `UIPopoverPresentationController`
  It notes what caused the popover to appear (a bar button item or just a rectangle in a view)
  You can also control what direction the popover's arrow is allowed to point
  Or you can control how a popover adapts to different sizes classes (e.g. iPad vs iPhone)

Popover Prepare

- Here's a prepare(for segue:) that prepares for a Popover segue

```
func prepare(for segue: UIStoryboardSegue, sender: Any?) {
    if let identifier = segue.identifier {
        switch identifier {
            case "Do Something in a Popover Segue":
                if let vc = segue.destination as? MyController {
                    if let ppc = vc.popoverPresentationController {
                        ppc.permittedArrowDirections = UIPopoverArrowDirection.any
                        ppc.delegate = self
                    }
                    // more preparation here
                }
            default: break
        }
    }
}
```

One thing that is different is that we are retrieving the popover's presentation controller

Popover Prepare

- Here's a prepare(for segue:) that prepares for a Popover segue

```
func prepare(for segue: UIStoryboardSegue, sender: Any?) {
    if let identifier = segue.identifier {
        switch identifier {
            case "Do Something in a Popover Segue":
                if let vc = segue.destination as? MyController {
                    if let ppc = vc.popoverPresentationController {
                        ppc.permittedArrowDirections = UIPopoverArrowDirection.any
                        ppc.delegate = self
                    }
                    // more preparation here
                }
            default: break
        }
    }
}
```

We can use it to set some properties that will control how the popover pops up

Popover Prepare

- Here's a `prepare(for segue:)` that prepares for a Popover segue

```
func prepare(for segue: UIStoryboardSegue, sender: Any?) {
    if let identifier = segue.identifier {
        switch identifier {
            case "Do Something in a Popover Segue":
                if let vc = segue.destination as? MyController {
                    if let ppc = vc.popoverPresentationController {
                        ppc.permittedArrowDirections = UIPopoverArrowDirection.any
                        ppc.delegate = self
                    }
                    // more preparation here
                }
            default: break
        }
    }
}
```

And we can control the presentation by setting ourself (the Controller) as the delegate

Popover Presentation Controller

- Adaptation to different size classes

One very interesting thing is how a popover presentation can "adapt" to different size classes.
When a popover is presenting itself in a horizontally compact environment (e.g. iPhone),
there might not be enough room to show a popover window comfortably,
so by default it "adapts" and shows the MVC in full screen modal instead.

But the popover presentation controller's delegate can control this "adaptation" behavior.
Either by preventing it entirely …

```
func adaptivePresentationStyle(
     for controller: UIPresentationController,
    traitCollection: UITraitCollection
) -> UIModalPresentationStyle {
    return UIModalPresentationStyle.none // don't adapt
    // the default in horizontally compact environments (iPhone) is .fullScreen
}
```

Popover Presentation Controller

- Adaptation to different size classes

  One very interesting thing is how a popover presentation can "adapt" to different size classes.
  When a popover is presenting itself in a horizontally compact environment (e.g. iPhone),
  there might not be enough room to show a popover window comfortably,
  so by default it "adapts" and shows the MVC in full screen modal instead.

  But the popover presentation controller's `delegate` can control this "adaptation" behavior.
  … or by modifying the adaptation …
  You can control the view controller that is used to present in the adapted environment
  Best example: wrapping a `UINavigationController` around the MVC that is presented

```
func presentationController(controller: UIPresentationController,
   viewControllerForAdaptivePresentationStyle: UIModalPresentationStyle)
    -> UIViewController?
{
    // return a UIViewController to use (e.g. wrap a Navigation Controller around your MVC)
}
```

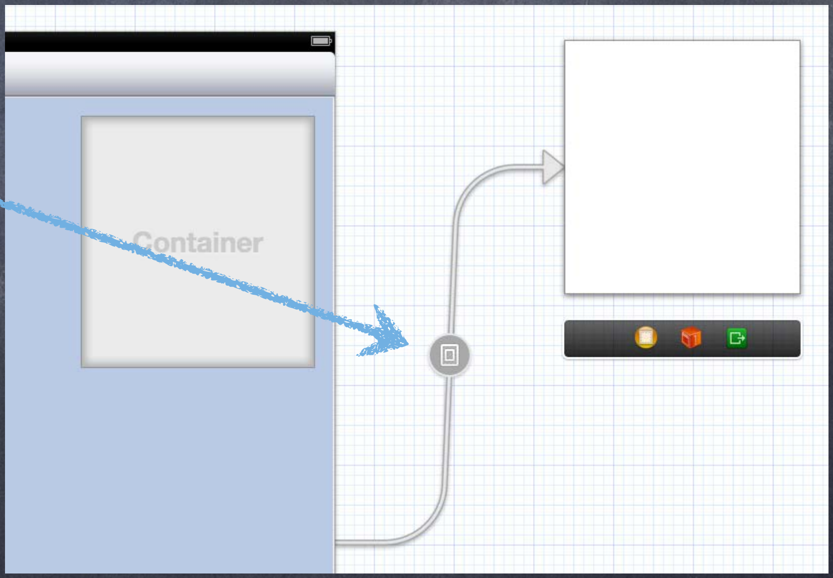
Popover Size

- Important Popover Issue: Size

  A popover will be made pretty large unless someone tells it otherwise.
  The MVC being presented knows best what it's "preferred" size inside a popover would be.
  It expresses that via this property in itself (i.e. in the Controller of the MVC being presented) …
  `var preferredContentSize: CGSize`
  The MVC is not guaranteed to be that size, but the system will try its best.
  You can set or `override` the `var` to always return an appropriate size.

Embed Segues

- Putting a VC's `self.view` in another VC's view hierarchy!
  This can be a very powerful encapsulation technique.
- Xcode makes this easy
  Drag out a Container View from the object palette into the scene you want to embed it in.
  Automatically sets up an "Embed Segue" from container VC to the contained VC.
- Embed Segue
  Works just like other segues.
  `prepare(for segue:, sender:)`, et. al.



Embed Segues

- Putting a VC's `self.view` in another VC's view hierarchy!
  - This can be a very powerful encapsulation technique.
- Xcode makes this easy
  - Drag out a Container View from the object palette into the scene you want to embed it in.
  - Automatically sets up an "Embed Segue" from container VC to the contained VC.
- Embed Segue
  - Works just like other segues.
  - `prepare(for segue:, sender:)`, et. al.
- View Loading Timing
  - Don't forget, though, that just like other segued-to VCs,
    the embedded VC's outlets are not set at the time `prepare(for segue:, sender:)` is called.

Demo

- Upgrading Emotions version of FaceIt
  - Use a Table View instead of hardwired Sad, Happy, etc.
  - Allow adding new emotions to this table (via a Modal segue)
  - Use an embed segue to make this UI even better
  - Unwind from this new Modal MVC back to our Emotions MVC
  - Make it look nice on iPad to with a popover
  - Make popover fit its contents better
  - Prevent adaptation in vertically compact environments