iOS Dev Accelerator Week2 Day3

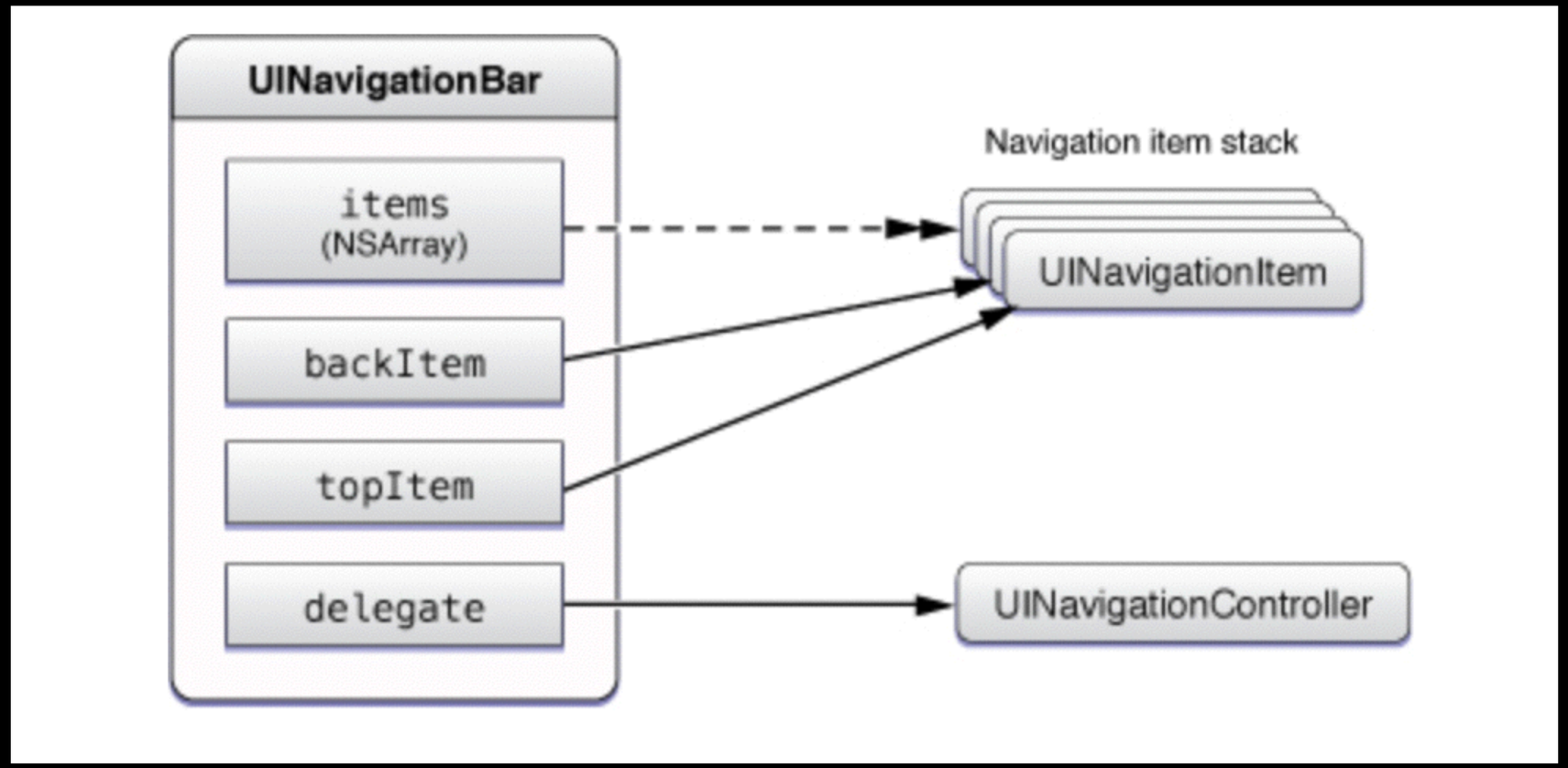
- UINavigationItem
- UIImagePickerController
- Photos Framework
- Social Sharing

Homework Review

UINavigationItem

Navigation Bar Anatomy

- A navigation bar has pretty similar setup as the navigation controller:



UINavigationItem

- UINavigationItem provides the content that the navigation bar displays. It is a wrapper object that manages the buttons and views to display in a navigation bar.
- The managing navigation controller uses the navigation items of the topmost two view controllers to populate the navigation bar with content.
- The navigation bar keeps a stack of all the items, in the exact same order as the navigation controller keeps track of its child content view controllers.
- Each View controller has a property that points to its corresponding navigation item
- The navigation bar has 3 positions for items: left, center, and right.

UINavigationItem positions

- Left: usually reserved for the back button, but you can replace it with whatever view you want by setting the navigation bar's leftBarButtonItem property.
- Center: Displays the title of the currently displayed view controller.
- Right: Empty by default, is typically used to place buttons that fire off actions.

Camera Programming

- 2 ways for interfacing with the camera in your app:
  1. UIImagePickerController (easy mode)
  2. AVFoundation Framework (hard mode)

UIImagePickerController

- The workflow of using UIImagePickerController is 3 steps:
    1. Instantiate and modally present the UIImagePickerController
    2. ImagePicker manages the user's interaction with the camera or photo library
    3. The system invokes your image picker controller delegate methods to handle the user being done with the picker.

UIImagePickerController Setup

- The first thing you have to account for is checking if the device has a camera.
- If your app absolutely relies on a camera, add a UIRequiredDeviceCapabilities key in your info.plist
- Use the isSourceTypeAvailable class method on UIImagePickerController to check if camera is available.

UIImagePickerController Setup

- Next make sure something is setup to be the delegate of the picker. This is usually the view controller that is spawning the picker.
- The final step is to actually create the UIImagePicker with a sourceType of UIImagePickerControllerSourceTypeCamera.
- Media Types: Used to specify if the camera should be locked to photos, videos, or both.
- AllowsEditing property to set if the user is able to modify the photo in the picker after taking the photo.

UIImagePickerControllerDelegate

- The Delegate methods control what happens after the user is done using the picker. 2 big method:
  1. imagePickerControllerDidCancel:
  2. imagePickerController:didFinishPickingMediaWithInfo:

Info Dictionary

The info dictionary has a number of items related to the image that was taken:

```
NSString *const UIImagePickerControllerMediaType;
NSString *const UIImagePickerControllerOriginalImage;
NSString *const UIImagePickerControllerEditedImage;
NSString *const UIImagePickerControllerCropRect;
NSString *const UIImagePickerControllerMediaURL;
NSString *const UIImagePickerControllerReferenceURL;
NSString *const UIImagePickerControllerMediaMetadata;
```

MediaType is either kUTTypeImage or kUTTypeMovie

Demo

Photos Framework

Photos Framework

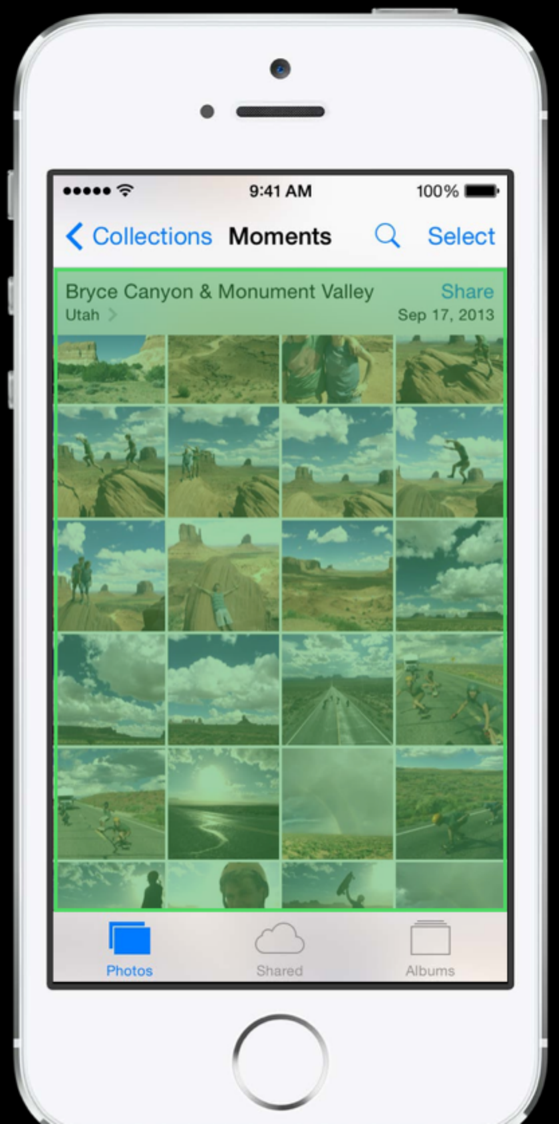
- New Apple Framework that allows access to photos and videos from the photo library.
- Also used for creating photo editing app extensions, a new feature with iOS8
- First-class citizen, you can create a full-featured photo library browser and editor on par with Apple's Photos App.
- Intended to supersede ALAssetsLibrary

PHAsset

- The Photos framework model object that represents a single photo or video.
- Has properties for Media type, Creation date, Location, and Favorite.

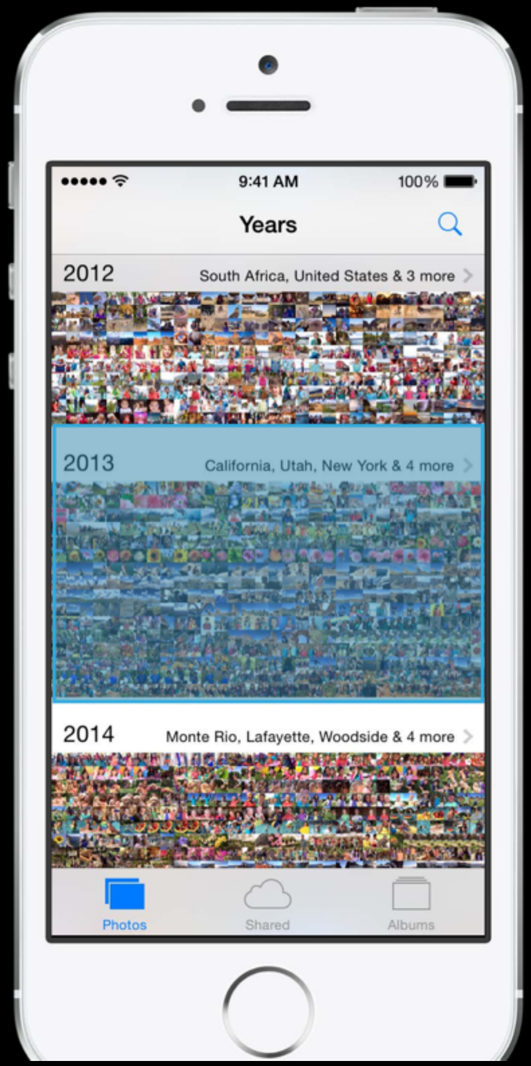
PHAssetCollection

- A Photos framework model object representing an ordered collection of assets.
- Albums, moments, and smart albums.
- Has properties for Type, Title, and Start and End Date.



PHCollectionList

- A Photos framework model object representing an ordered collection of collections.
- This can be a folder, moment, or year
- Has properties for Type, Title, and Start and End Date.



Fetching Model Objects

- You fetch via class methods on the models:
  - PHAsset.fetchAssetsWithMediaType(PHAssetMediaType.Photo, options:nil)
  - PHAssetCollection.fetchMomentsWithOptions(nil)
- Collections do not cache their contents, so you still have to fetch all the assets inside of it. This is because the results of a fetch can be very large, and you dont want all of those objects in memory at once.

PFFetchResult

- Results returned in a PHFetchResult
- Similar to an Array.

Making Changes

- You can favorite a photo and add an asset to an album
- You cannot directly mutate an asset, they are read only (thread safe!)
- To make a change, you have to make a change request.
- Theres a request class for each model class:

```
PHAssetChangeRequest
PHAssetCollectionChangeRequest
PHCollectionListChangeRequest
```

Making Changes

```
func toggleFavorite(asset : PHAsset) {

    PHPhotoLibrary.sharedPhotoLibrary().performChanges({

        //create a change request object for the asset
        var changeRequest = PHAssetChangeRequest(forAsset: asset) as
      PHAssetChangeRequest
        //make your change
        changeRequest.favorite = !changeRequest.favorite

        }, completionHandler: { ( success : Bool,error : NSError!) -> Void in

        //asset change complete
        })

}
```

Making New Objects

Create via creation request

```
var request = PHAssetChangeRequest.creationRequestForAssetFromImage(UIImage())
```

Placeholder objects

```
var placeHolder = request.placeholderForCreatedAsset
```

- Reference to a new, unsaved object
- Add to collections
- Can provide unique, persistent `localIdentifier`

Getting to the actual data

- Many different sizes of an image may be available or different formats of a video
- Use PHImageManager to request images/videos
- Request an image based on target size for displaying
- Request a video based on the usage
- Asynchronous API, because you dont know how long it will take to load the data, it could be very expensive
- Will optionally retrieve the data from the network if its only on iCloud
- Use a PHCachingImageManager when displaying a collection of images for better performance.

Requesting an Image

```
let manager = PHImageManager.defaultManager()

manager.requestImageForAsset(photo,
                             targetSize: cellSize,
                             contentMode: PHImageContentMode.AspectFill,
                             options: nil,
                             resultHandler: {(result : UIImage!, [NSObject : AnyObject]!) -> Void in
                                 if result {
                                     imageView.image = result
                                 } else {
                                     //tell user something went wrong
                                 }
    })
```

Advanced Image Request

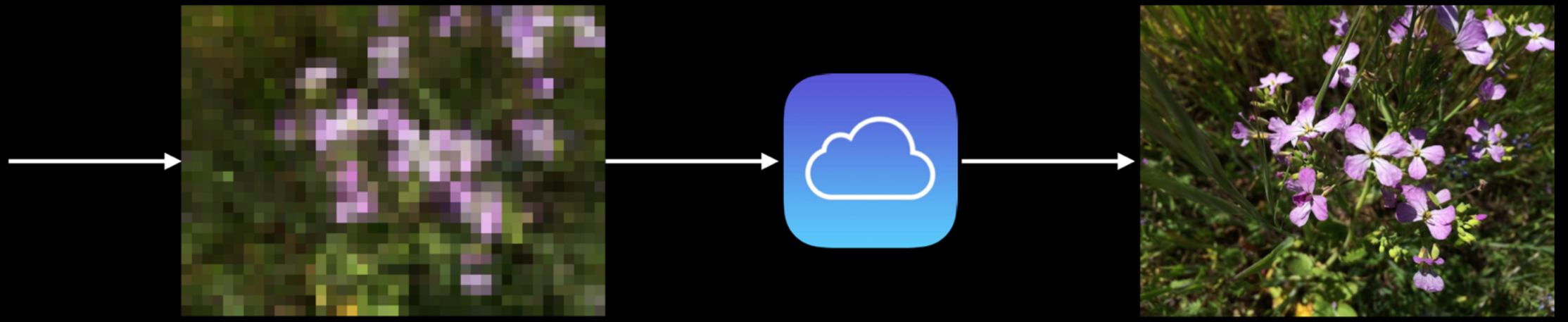
```
var options = PHImageRequestOptions()

options.networkAccessAllowed = true
options.progressHandler = {(progress : Double, error : NSError!, degraded : UnsafePointer<ObjCBool>,
[NSObject : AnyObject]!) in
    //update visible progress UI
    }

//use your options to control the request behavior
manager.requestImageForAsset(photo,
    targetSize: cellSize,
    contentMode: PHImageContentMode.AspectFill,
    options: options,
    resultHandler: {(result : UIImage!, [NSObject : AnyObject]!) -> Void in
```

Advanced Image Request

```
[manager requestImageForAsset: ... ^(UIImage *result, NSDictionary *info) {
    // This block can be called multiple times
}];
```



First callback synchronous

Second callback asynchronous

Demo

Social Sharing

SLComposeController

- SLComposerController class presents a view to the user to compose a post for the supported social networking services
- First check if the service type(s) you are going to offer are available on the user's device (aka they are signed in) by calling isAvailableForServiceType
- The available service types are facebook,twitter, weibo, and tencentWeibo.

SLComposeController Workflow

1. Check if the service type is available
2. instantiate an SLCompViewController object, and use the init that takes in a SLServiceType
3. Add whatever image or URL you are going to share if you have them.
4. Add a completionHandler of type (SLComposeViewControllerResult) -> (Void)
5. Present the view controller

Demo