lecture 9

AVFoundation and Location

cs198-001 : fall 2017

Announcements

- Snapchat Clone Part 2 (hw3 pt2) due tonight (11:59pm)
  - remember - you have 3 slip days for part 1 and part 2

You will need an iPhone / iPad with iOS 11 and iPhone cable for lab this week!

(if you don’t have a device, you can work with a partner, as per usual)

Overview : Today's Lecture

Core Location

Map Kit

AVFoundation

Core Location

Review: Stuff + GPS

GPS drains battery and is unreliable in dense urban and indoor environments

Need accurate location services…

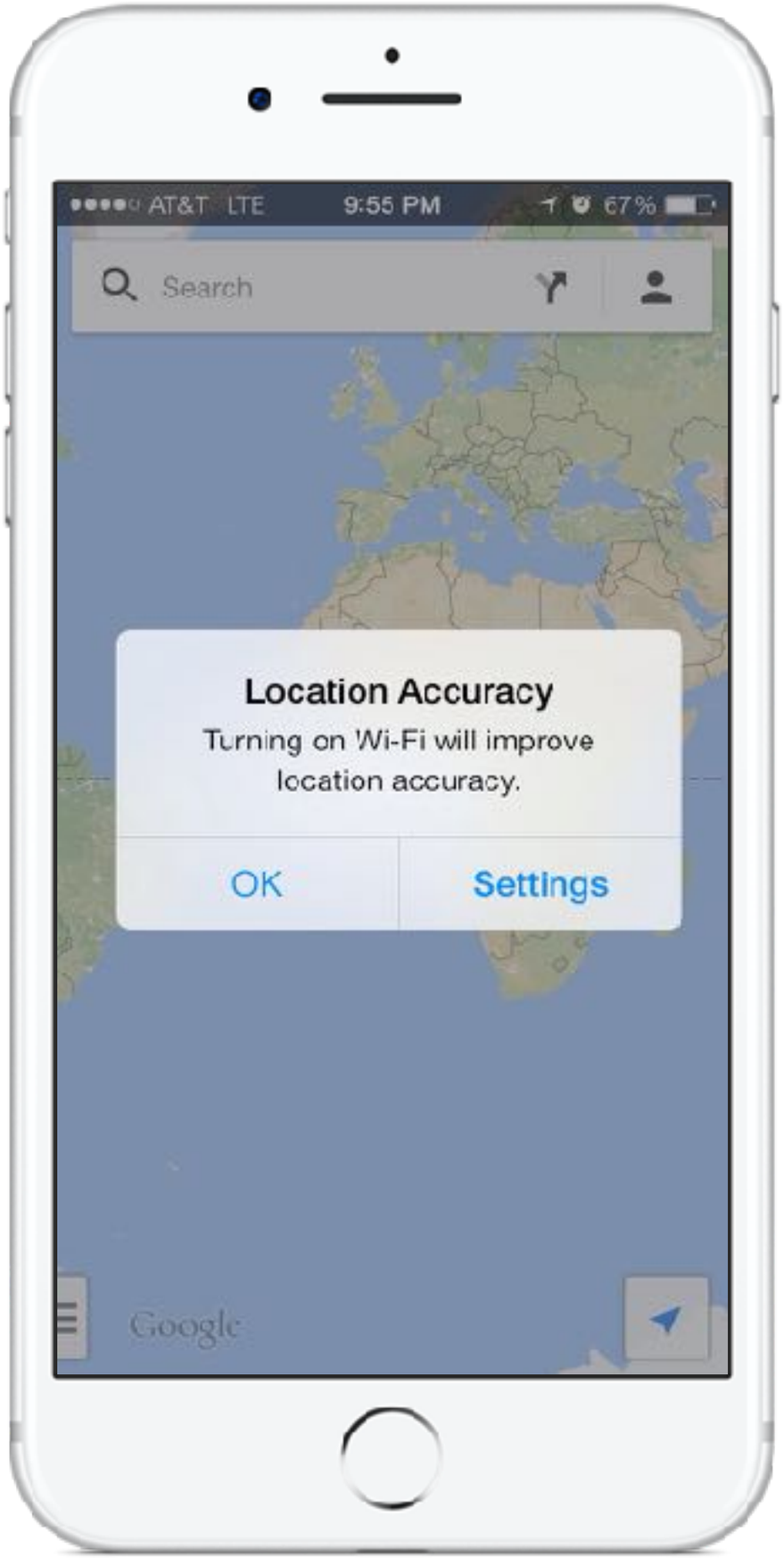
What can you do?

Review: Stuff + GPS

Hybrid Positioning System

Use crowd-sourced database of Wi-Fi hotspot and cell tower locations

= Core Location



Core Location : Configuration

Add the `NSLocationWhenInUseUsageDescription` key to your app's Info.plist file. ("Privacy - Location When In Use Usage Description")

Information Property List		Dictionary	(21 items)
Localization native development region	▲▼	String	United States
Bundle display name	▲▼	String	Outfit
Executable file	▲▼	String	${EXECUTABLE_NAME}
Get Info string	▲▼	String	
Bundle identifier	▲▼	String	com.codebrew.poker
InfoDictionary version	▲▼	String	6.0
Bundle name	▲▼	String	${PRODUCT_NAME}
Bundle OS Type code	▲▼	String	APPL
Bundle versions string, short	▲▼	String	1.0
Bundle creator OS Type code	▲▼	String	????
▶ URL types	▲▼	Array	(1 item)
Bundle version	▲▼	String	1.0
FacebookAppID	▲▼	String	709337579107040
FacebookDisplayName	▲▼	String	Photo Collage Maker
Application Category	▲▼	String	
Application requires iPhone environment	▲▼	Boolean	YES
View controller-based status bar appearance	▲▼	Boolean	NO
Copyright (human-readable)	▲▼	String	
▶ Required device capabilities	▲▼	Array	(1 item)
▶ Supported interface orientations	▲▼	Array	(2 items)
NSLocationWhenInUseUsageDescription	▲▼ ⊕ ⊖	String	▲▼ This application requires location services to work.

Core Location : Configuration

OR both:

- `NSLocationWhenInUseUsageDescription`
- `NSLocationAlwaysAndWhenInUseUsageDescription`

keys to your app's Info.plist file.

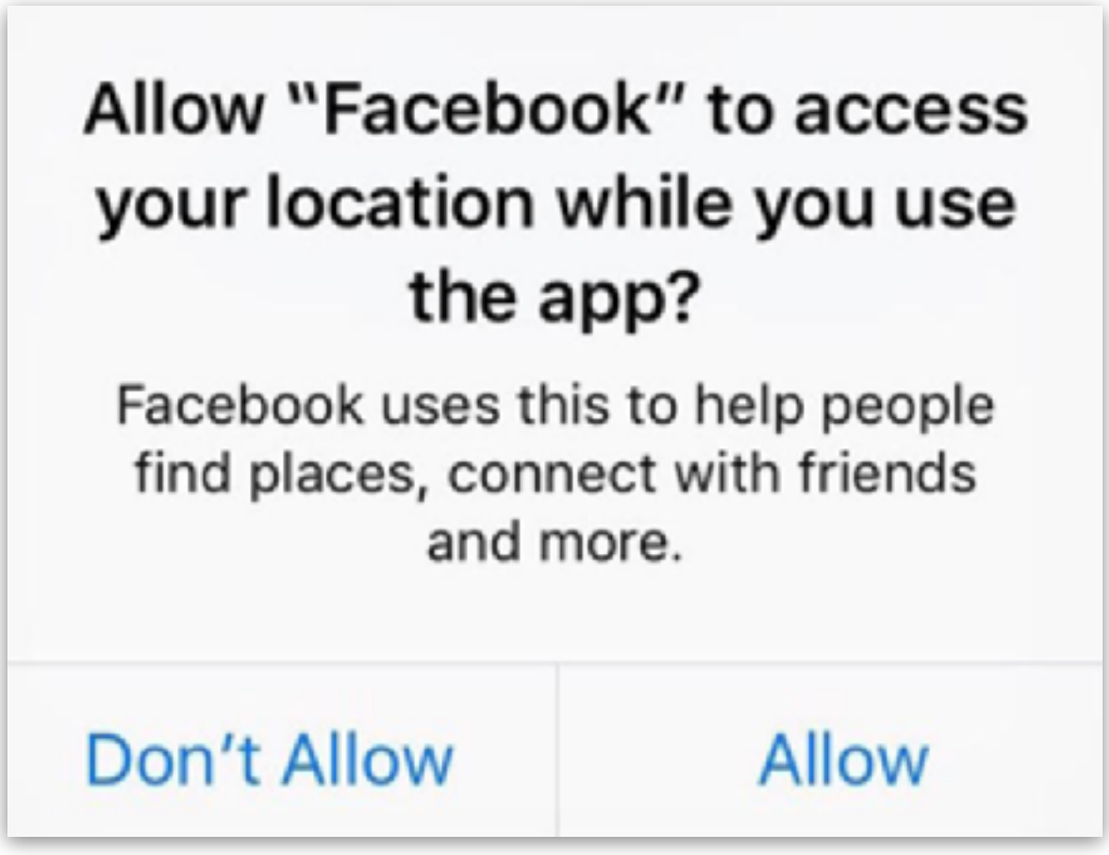
Core Location : Permissions

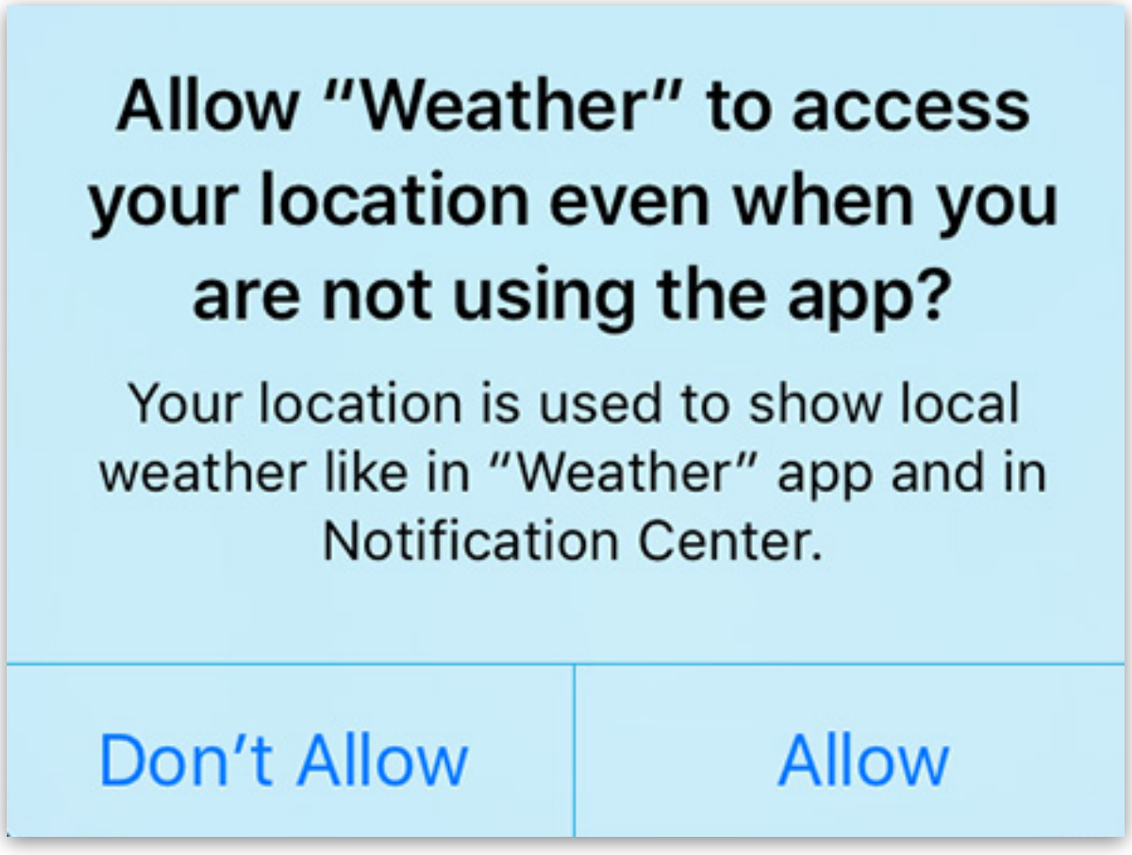
Before getting a user’s location, they have to have enabled Location Services for your app

```
let manager = CLLocationManager()

if !CLLocationManager.locationServicesEnabled()
{
    //ask for user’s location
}
```

Core Location : Permissions

When In Use vs. Always






```
manager.requestWhenInUseAuthorization()
```

```
manager.requestAlwaysAuthorization()
```

Core Location : Permissions

Let’s say you always want the user’s location

Even when not in the app (background)

```
switch CLLocationManager.authorizationStatus() {
      case .authorizedAlways:
          break
       case .notDetermined:
          manager.requestAlwaysAuthorization()
       case .authorizedWhenInUse, .restricted, .denied:
          //prompt notification: see next slides
}
```

Alerts

```
let alertController = UIAlertController(
            title: "Background Location Access
Disabled",
        message: "In order to ____, please open
                Settings and set location access
                for this app to 'Always'.",
        preferredStyle: .alert)
```

Alerts

```
let openAction = UIAlertAction(title: "Open Settings",
                               style: .default)
{ (action) in
    if let url = NSURL(string:
UIApplicationOpenSettingsURLString) {
        UIApplication.shared.open(url as URL,
                                  options: [:],
                                  completionHandler: nil)
    }
}

alertController.addAction(openAction)
```

Alerts

```
let cancelAction = UIAlertAction(title:
"Cancel", style: .cancel, handler: nil)


alertController.addAction(cancelAction)
```

Alerts

To actually present the alert in your desired context...

```
self.present(alertController,
animated: true,
completion: nil)
```

Core Location : One Time Location

Fetch user's location once

```
let manager = CLLocationManager()

override func viewDidLoad() {
    super.viewDidLoad()
    // manager is your CLLocationManager
    manager.delegate = self //important!!
    manager.desiredAccuracy =
    kCLLocationAccuracyBest
    manager.requestLocation() //type of update
}
```

Core Location : One Time Location

Calling the method

```
manager.requestLocation()
```

Will call either:

```
locationManager(_:didUpdateLocations:)
locationManager(_:didFailWithError:)
```

in your `CLLocationManagerDelegate` class (that's why you must set the delegate!)

Core Location : Location over Time

Standard Location Service

For continuous updates (e.g. Maps)

```
manager.startUpdatingLocation()
```

Significant-Change Loc. Service

Update only when location changes

```
manager.startMonitoringSignificantLocationChanges()
```

Core Location : Location over Time

Must implement appropriate delegate method(s) in View Controller to receive data

```
func locationManager(_ manager: CLLocationManager,
                     didUpdateLocations locations:
                                          [CLLocation]) {
    // most recent location update at the end of the array
    let latestLocation = locations.last!
    // do something
}
```

Core Location : CLVisit

A period of time a user has spent in a single location, including both a coordinate and start/end timestamps

```
// initiating visit event updates
manager.startMonitoringVisits()
manager.stopMonitoringVisits()


// receive data via delegate method(s)
func locationManager(manager: CLLocationManager,
                     didVisit visit: CLVisit) {
}
```

Core Location : CLRegion

Monitor boundary crossings for defined geographical regions (geofencing)

```
// define desired geographical region and radius
let currRegion = CLCircularRegion(center:
                 CLLocationCoordinate2D(latitude: 37,
                                        longitude: 122),
                                 radius: 200,
                                 identifier: "Berkeley")

// initiating region monitoring
manager.startMonitoring(for: currRegion)
manager.stopMonitoring(for: currRegion)
```

Core Location : CLRegion

Monitor boundary crossings for defined geographical regions (geofencing)

```
//delegate method fires when user enters
func locationManager(_ manager: CLLocationManager,
didEnterRegion region: CLRegion) { … }


//delegate method fires when user exits
func locationManager(_ manager: CLLocationManager,
didExitRegion region: CLRegion) { … }
```

Core Location : Other

CLFloor - get information about what floor your user is on (returns int for floor)

iBeacons: developer.apple.com/ibeacon/

And more...

Core Location : User Trust

Keep location data secure

Do not auto-track user

Only use Location Services when they are needed

Map Kit

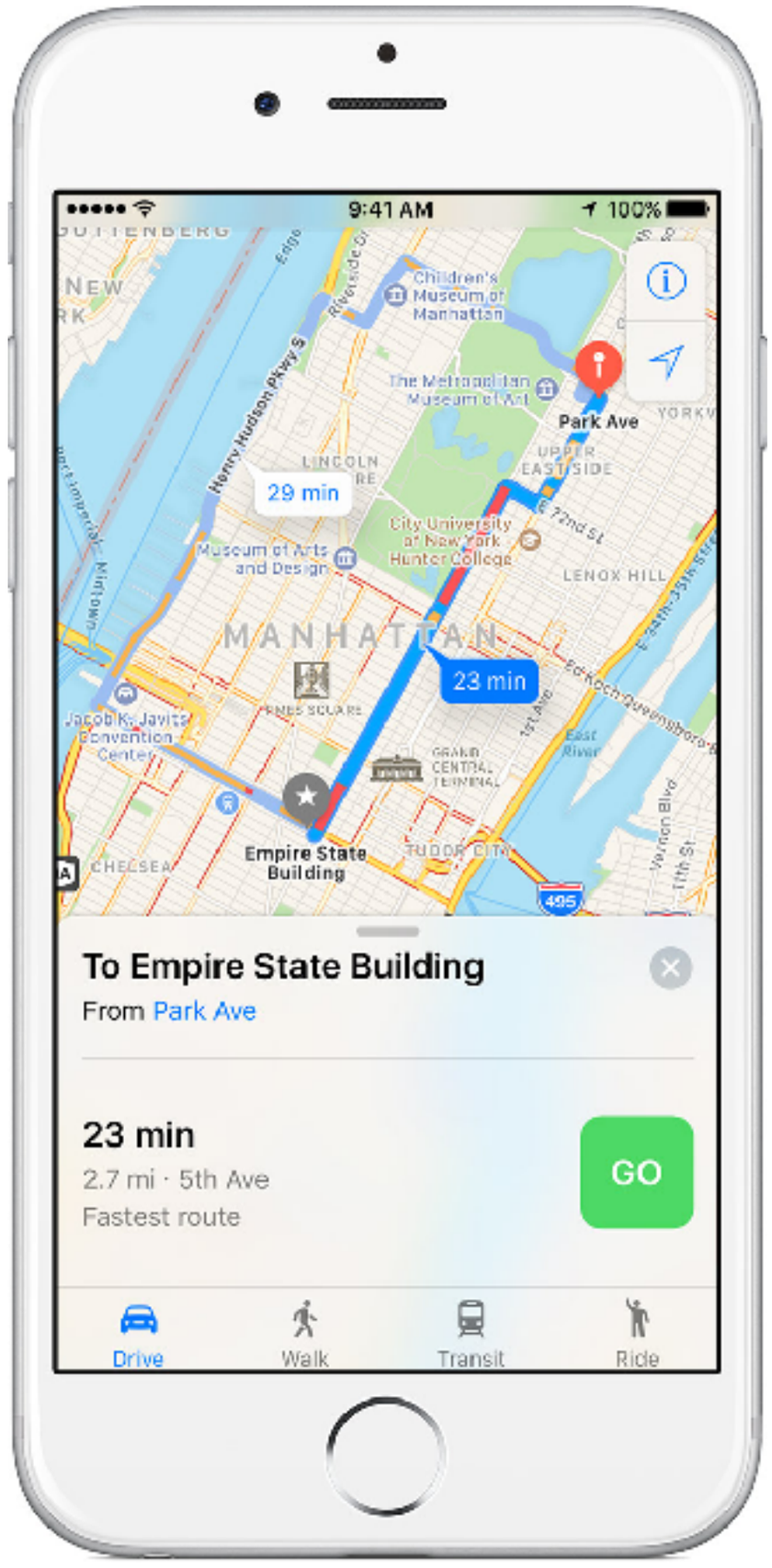
MapKit - Overview

API built off of CoreLocation

Embed maps directly to windows or views

Some Features:

- Annotate Map & Add Overlays
- Plot Location
- Jump to coordinates



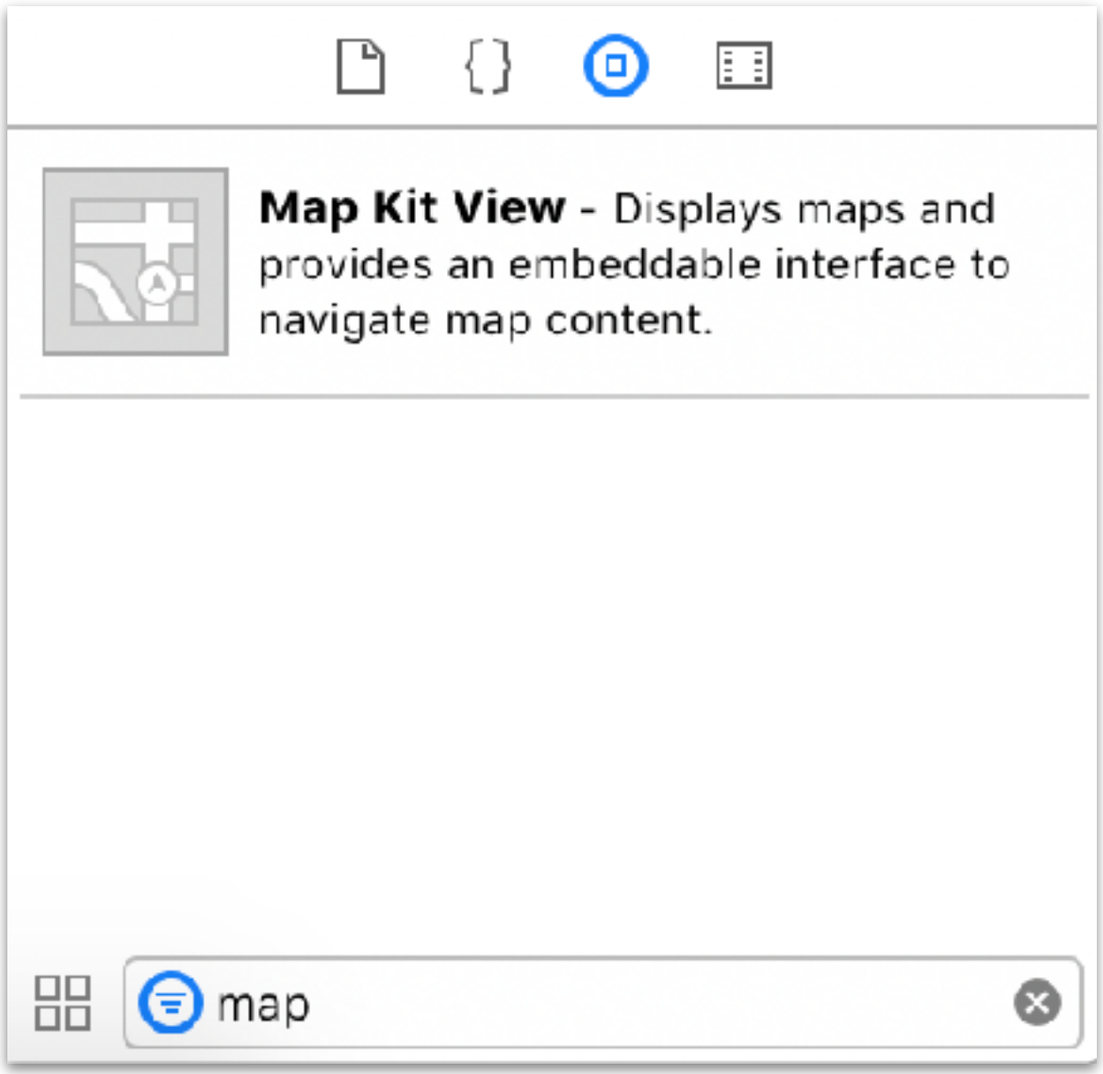
MapKit Example

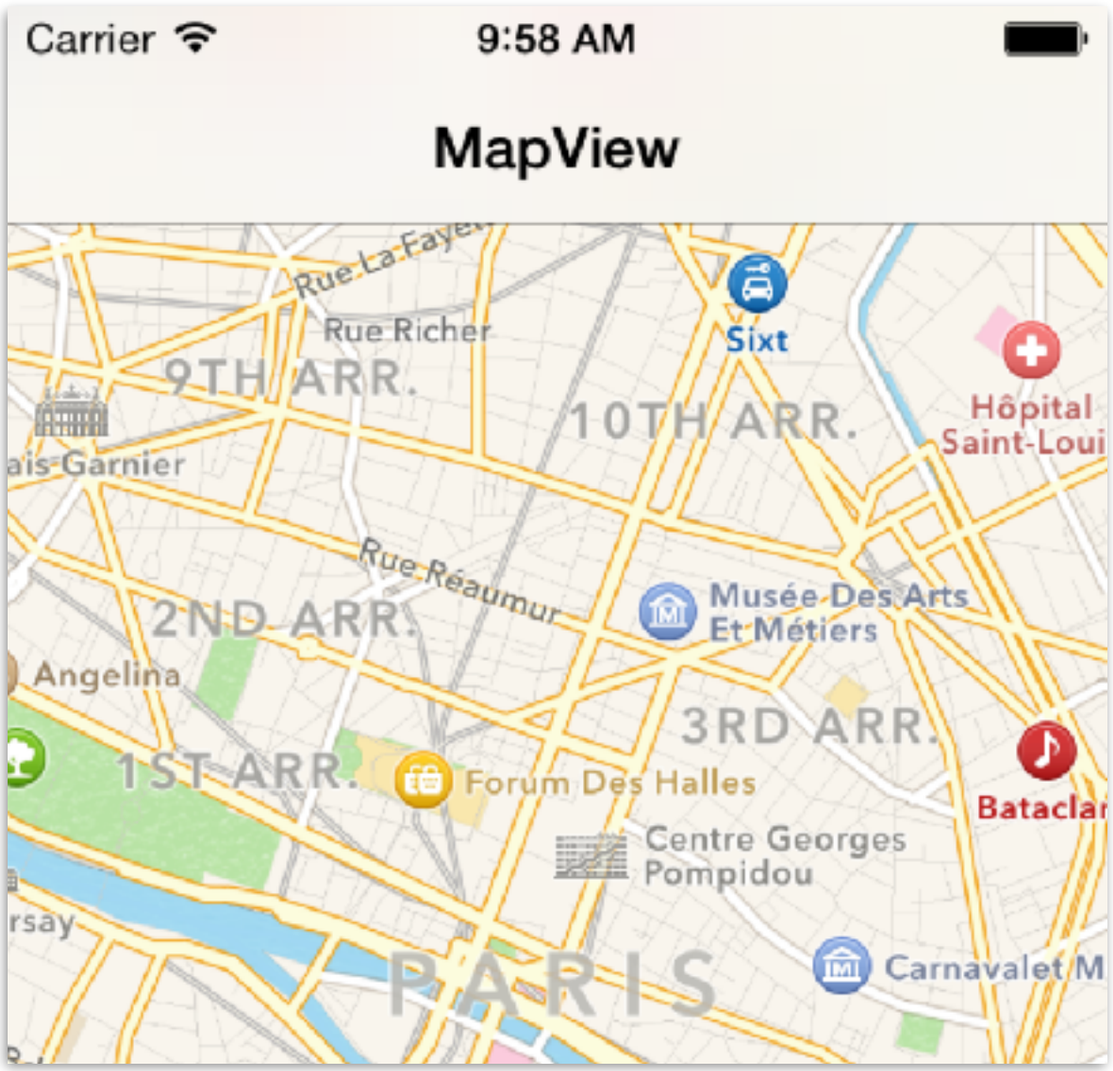
Easily embed an interactive map within your application with annotations

Example: Yelp

Embedding a Map : Storyboard

Drag a “Map Kit View” from the Object Library into your View Controller.





Custom Initial Map View

```
override func viewDidLoad() {
    super.viewDidLoad()
    let initialLoc = CLLocation(latitude:38,
                                longitude:122)
    centerMapOnLocation(initialLoc) //custom
}
```

Custom Initial Map View

```
// in your map's view controller
let regionRadius: CLLocationDistance = 1000

func centerMapOnLocation(location: CLLocation)
{
    let coordinateRegion =
       MKCoordinateRegionMakeWithDistance(
            location.coordinate,
            regionRadius * 2.0,
            regionRadius * 2.0)
    mapView.setRegion(coordinateRegion,
                      animated: true)
}
```

Adding Annotations

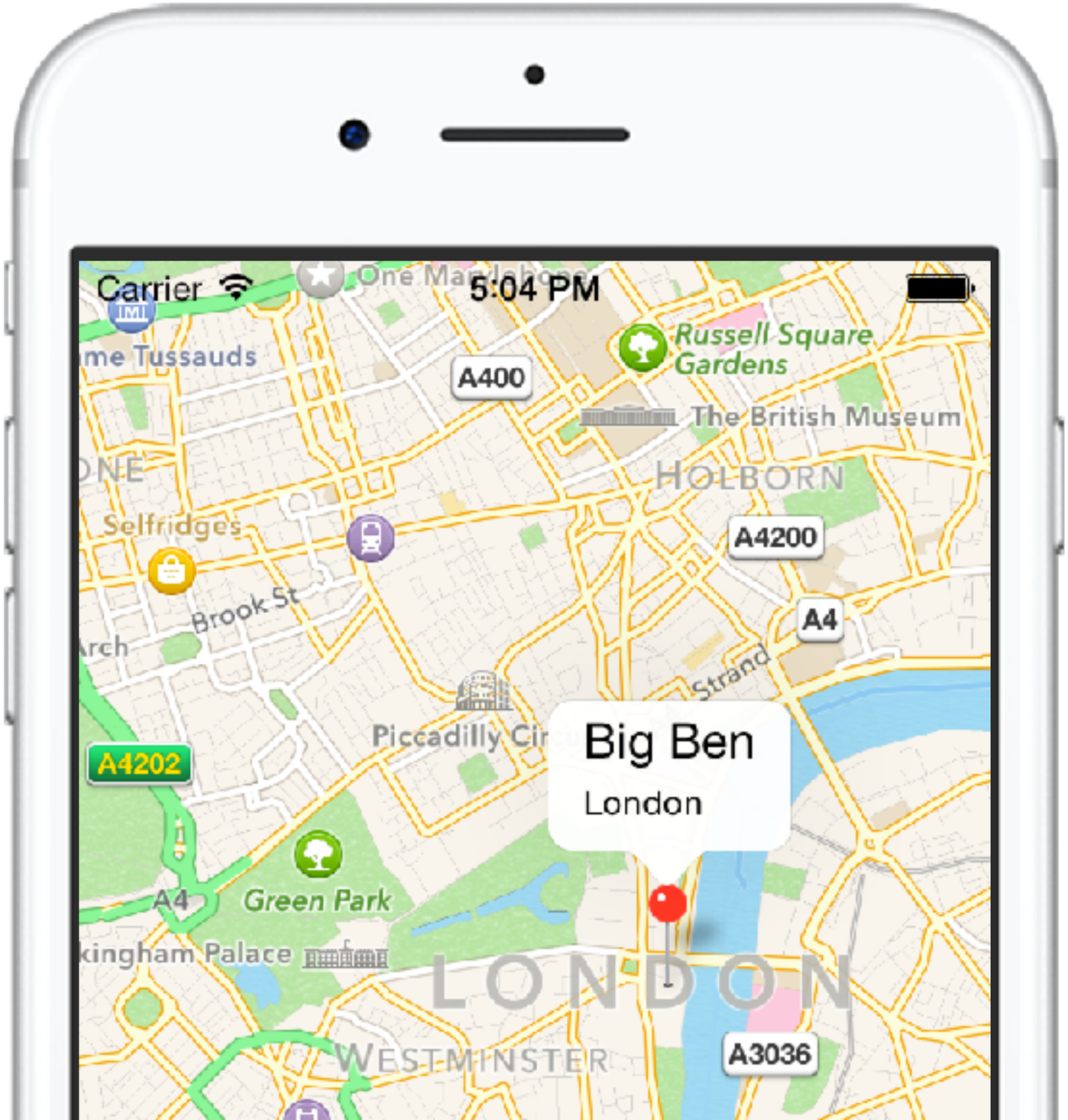
```
override func viewDidLoad() {
    let annotation = MKPointAnnotation()
    annotation.coordinate =
            CLLocationCoordinate2D(
                        latitude: 24,
                        longitude: 54)
    annotation.title = "Big Ben"
    annotation.subtitle = "London"
    mapView.addAnnotation(annotation)
}
```

What we have done so far

Created a Initial Map View

Set a location

Added an Annotation (with a title, subtitle, and coordinated)



Check In

AVFoundation

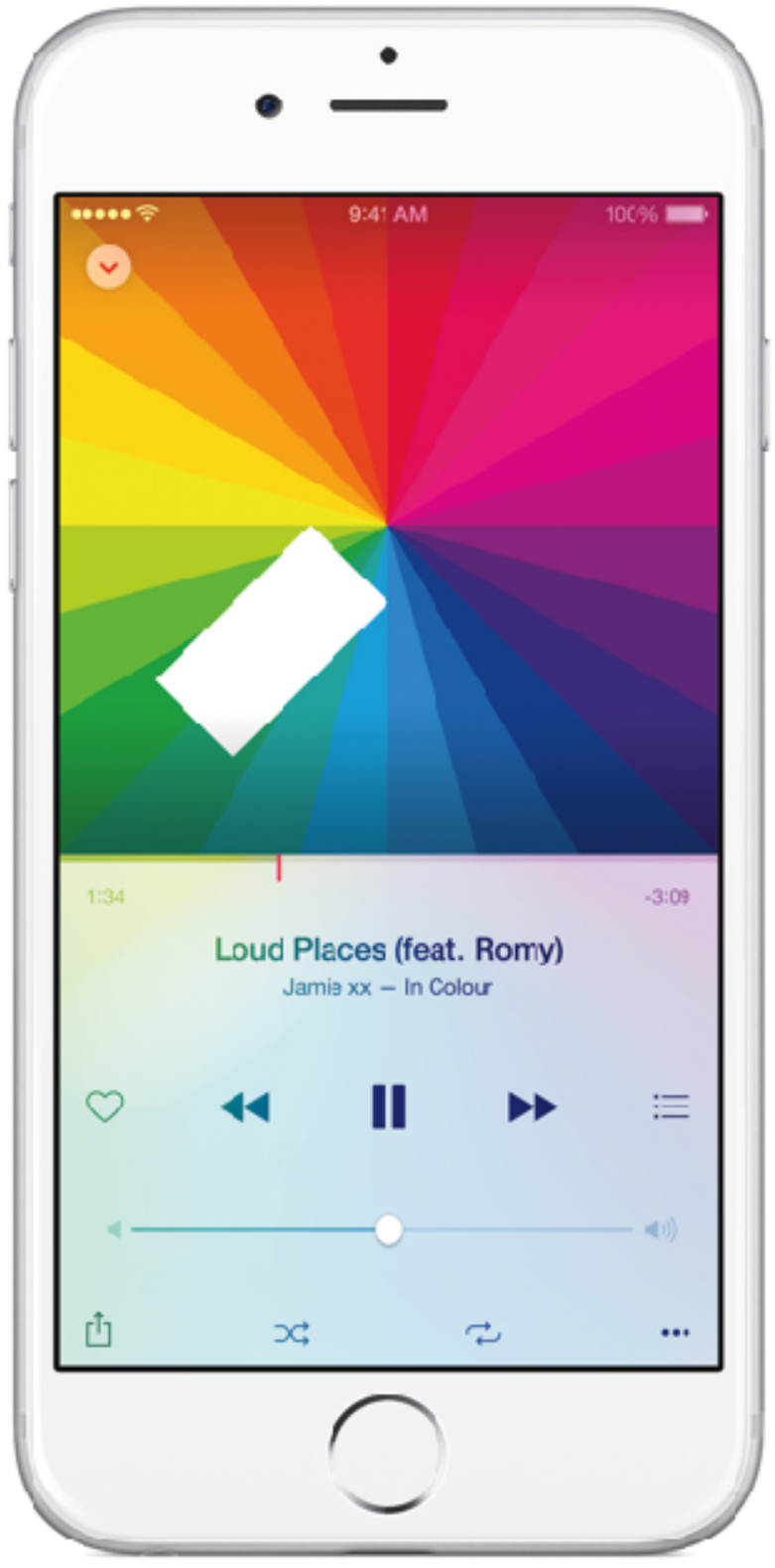
AVPlayer - Overview

Play audio in your App

Create an **AVAudioPlayer** to play your audio on

Create an **AVPlayerItem** for each sound clip / song

Use the player to play, pause, rewind, and fast forward your **AVPlayerItems**



Apple Music

AVPlayerItem - Initialization

Create an **AVPlayerItem** for each song / sound you want played.

Each **AVPlayerItem** is a single instance being played by AVPlayer

```
let item = AVPlayerItem(URL: someURL!)
```

AVPlayer - Initialization

Initialize an **AVPlayer** with or without a Player Item. You will add items to the player, then use the player to play these items.

```
let player = AVPlayer()

let player = AVPlayer(playerItem:
item)
```

AVPlayer - Playback

Once you've added some AVPlayerItems to your player, you can play, pause, fast forward, replace, etc.

```
let player = AVPlayer(playerItem:
item)

player.play()
player.pause()
player.seek(to: <CMTime>)
player.replaceCurrentItem(with:
newSong)
```

AVPlayer Example

```
import AVFoundation
...

func playSongFromURL(songURL: URL) {
   let song = AVPlayerItem(url: songURL)
   let player = AVPlayer(playerItem: song)
   if (player.currentItem!.status
   == .readyToPlay) {
      player.play()
   }
}
```

AVPlayer Example

```
import AVFoundation
...

func playSongFromURL(songURL: URL) {
   let song = AVPlayerItem(url: songURL)
   let player = AVPlayer(playerItem: song)
   if (player.currentItem!.status
   == .readyToPlay) {
      player.play()
   }
}
```

Import the **AVFoundation framework**
at the top of your file

AVPlayer Example

```
import AVFoundation
...

func playSongFromURL(songURL: URL) {
   let song = AVPlayerItem(url: songURL)
   let player = AVPlayer(playerItem: song)
   if (player.currentItem!.status
   == .readyToPlay) {
      player.play()
   }
}
```

Create an **AVPlayerItem** from a url or file in your application

AVPlayer Example

```
import AVFoundation
...

func playSongFromURL(songURL: URL) {
   let song = AVPlayerItem(url: songURL)
   let player = AVPlayer(playerItem: song)
   if (player.currentItem!.status
   == .readyToPlay) {
      player.play()
   }
}
```

Add that **AVPlayerItem** to an **AVPlayer** (here, we are initializing the **AVPlayer** with the item)

AVPlayer Example

```
import AVFoundation
...

func playSongFromURL(songURL: URL) {
   let song = AVPlayerItem(url: songURL)
   let player = AVPlayer(playerItem: song)
   if (player.currentItem!.status
   == .readyToPlay) {
      player.play()
   }
}
```

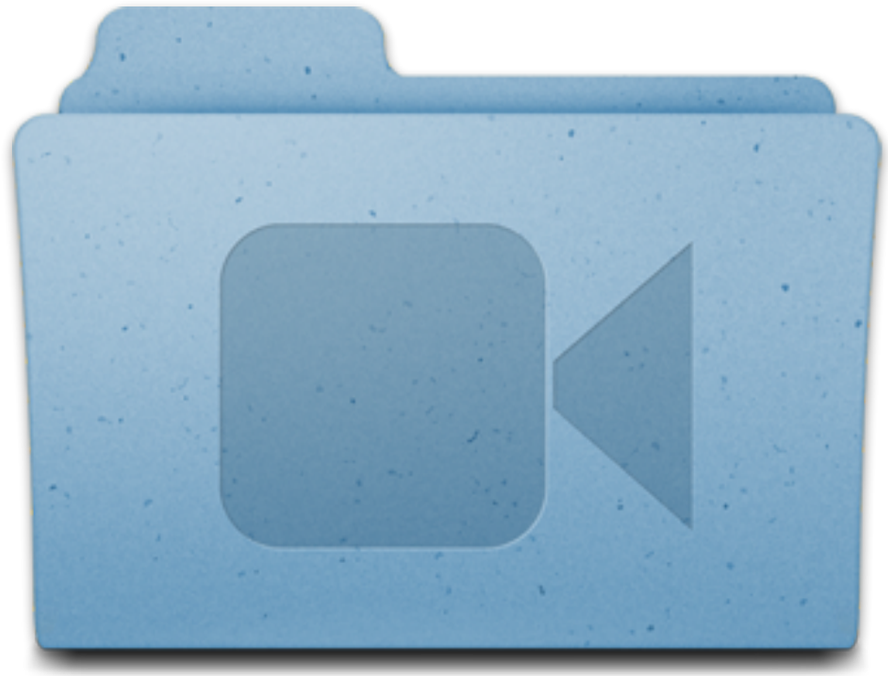
Now you can play, pause, seek, etc.

AVFoundation - What is it?

Cocoa framework for AudioVisual items

Used to record, edit, and stream media

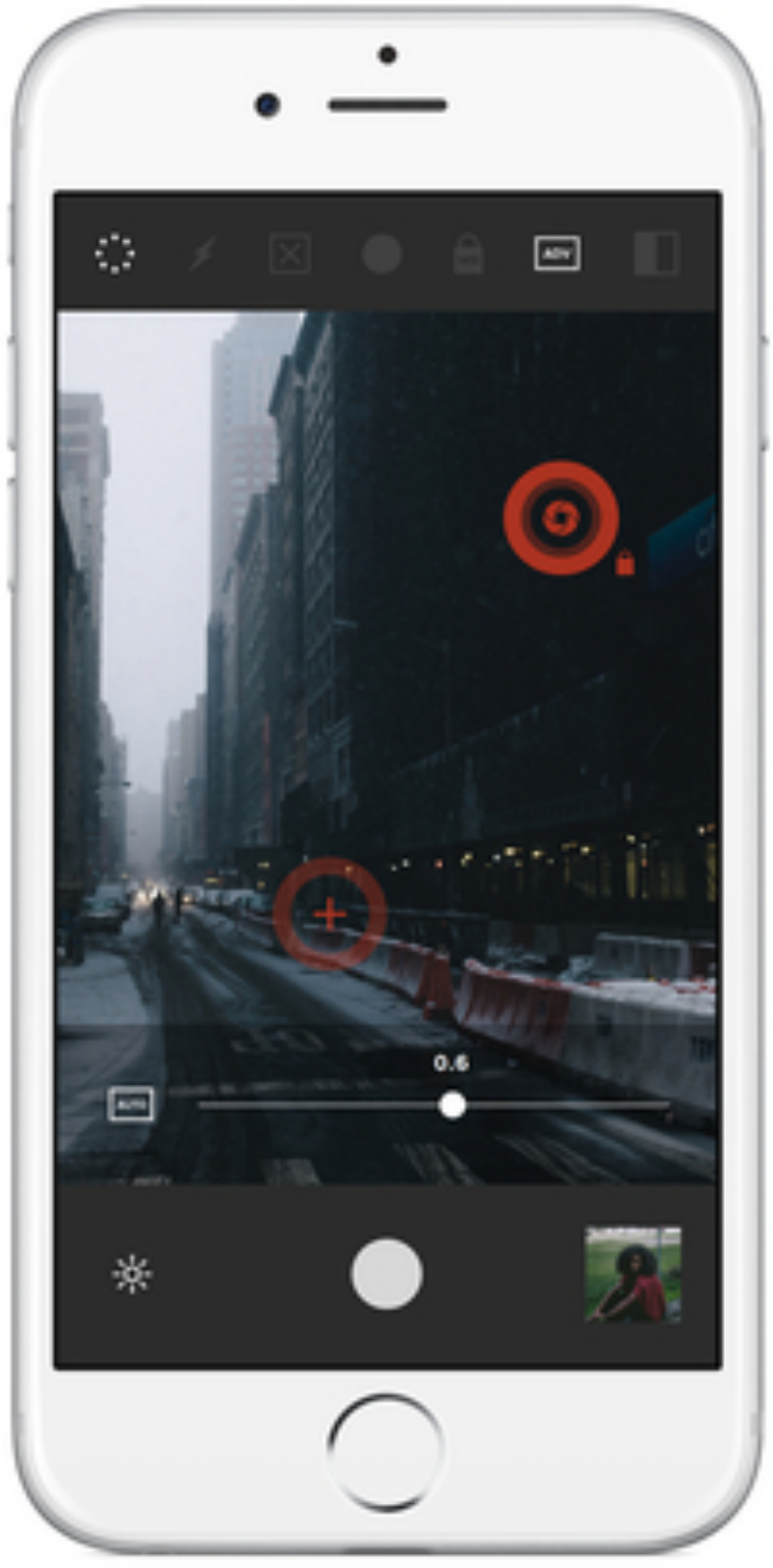
Includes Players, Items, ViewControllers, etc



AVCapture - Overview

Allows you to capture video, photo, scan barcodes, capture audio etc.

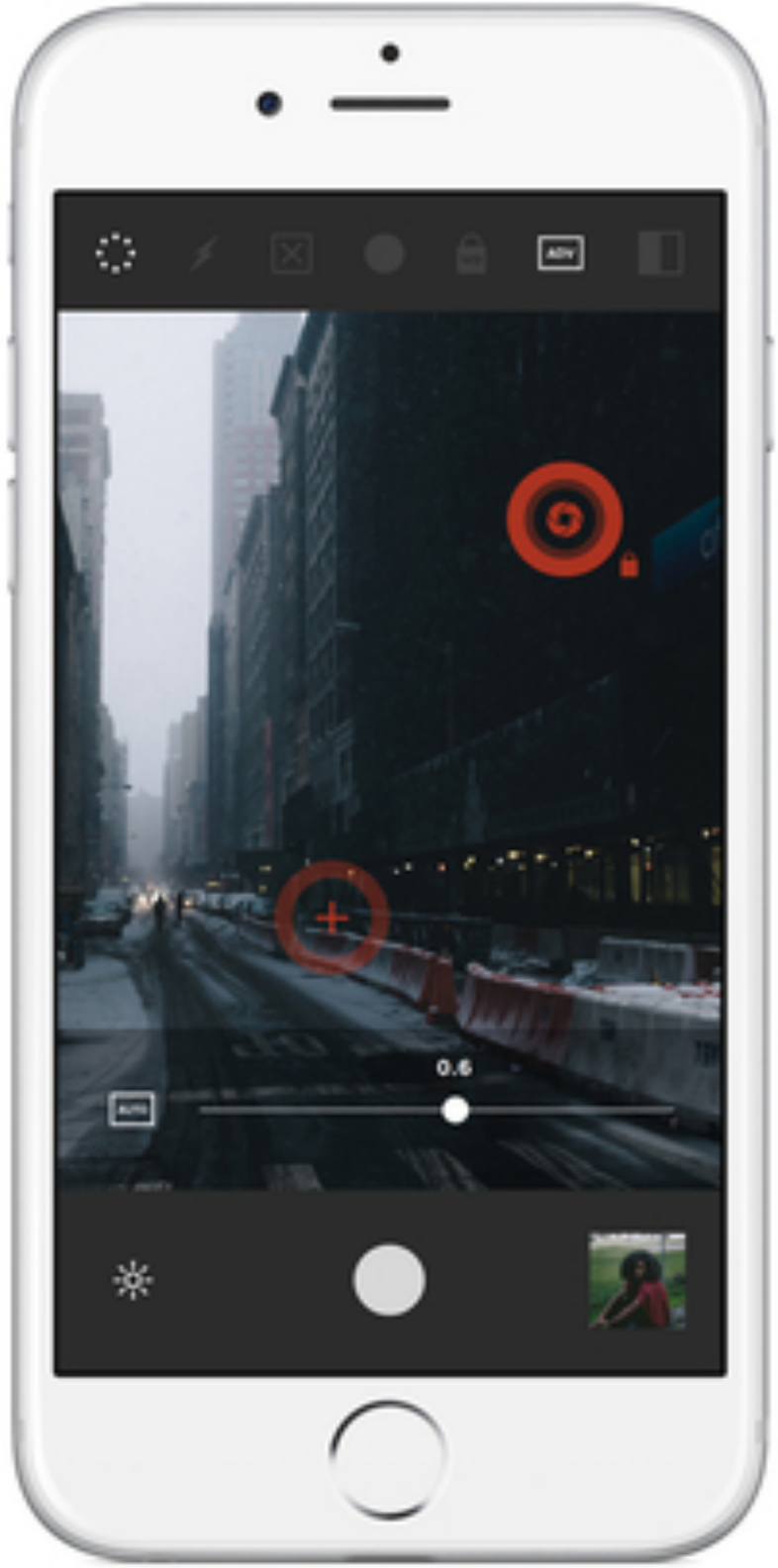
Note: If you just need to capture photo and video without custom formatting, use the UIKit framework instead (`UIImagePickerController`)



VSCO

to capture AV data….

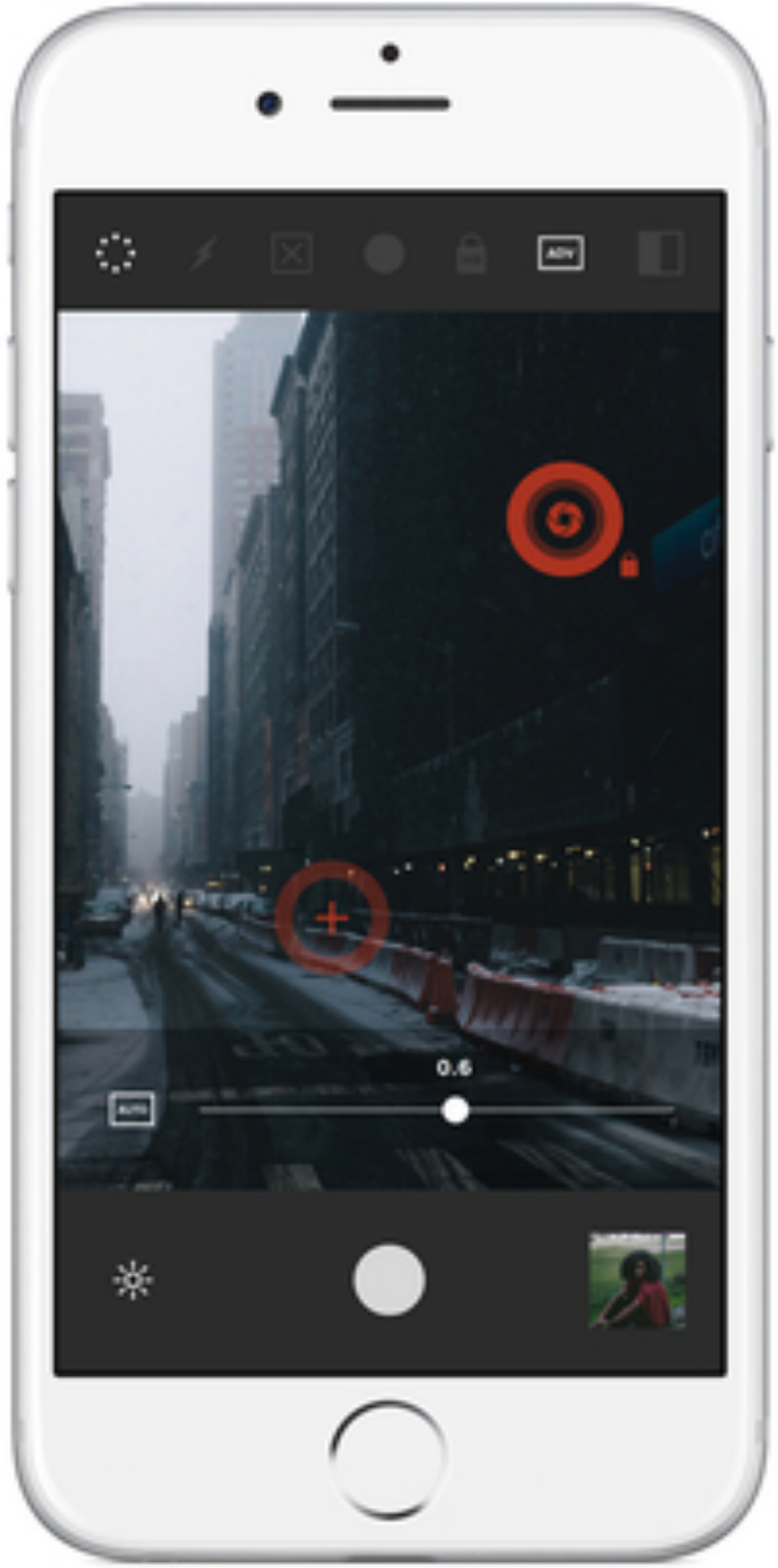
- Create a **AVCaptureSession**
- Set the **AVCaptureDeviceInput** depending on what you want to capture (video, photo, audio). Use AVCaptureDevice's DiscoverySession to get available inputs
- add outputs save data
- Begin media capture by calling `startRunning()` on your session
- use outputs to capture data from the session



VSCO

to capture AV data….

- Create a **AVCaptureSession**
- Set the **AVCaptureDeviceInput** depending on what you want to capture (video, photo, audio). Use AVCaptureDevice's DiscoverySession to get available inputs
- add outputs save data
- Begin media capture by calling `startRunning()` on your session
- use outputs to capture data from the session



VSCO

AVCaptureSession - creation

Creating a capture session

```
let captureSession = AVCaptureSession()

// set what quality you want to capture
captureSession.sessionPreset =
    AVCaptureSession.Preset.high
```

AVCaptureSession - adding Inputs

Find available “devices” using discovery sessions. In this case, we are filtering ones with a wide angle camera

```
let deviceDiscoverySession =
AVCaptureDevice.DiscoverySession(deviceTypes:
[.builtInWideAngleCamera], mediaType: .video,
position: .unspecified)

// devices will be of type [AVCaptureDevice]
let devices = deviceDiscoverySession.devices
```

AVCaptureSession - getting a device for input

Use devices as captureSession inputs

```
let input = try
AVCaptureDeviceInput(device: device)
```

```
captureSession.addInput(input)
```

Now you’re prepared to receive data (but not save) from camera or microphone, depending on what device you are using

AVCaptureSession - getting a device for input

Use devices as captureSession inputs

```
let input = try
AVCaptureDeviceInput(device: device)
```

```
captureSession.addInput(input)
```

Now you’re prepared to receive data (but not save) from camera or microphone, depending on what device you are using

but how do we access that data?? —> AVCaptureOutputs

AVCapture - Classes for output (photo)

AVCapturePhotoOutput - class used for outputting photos captured by device camera (still and live)

AVCapturePhotoSettings - pass along into AVCapturePhotoOutput object to set capture settings

- can specify what type to output (JPEG, RAW formats, HEIF, etc.), whether you should use flash, or if you should enable image stabilization.

AVCapturePhotoCaptureDelegate - responsible for receiving the data captured from the camera. This is where you can access the captured image

AVCapture - capturing photos

To actually capture a photo from the device camera:

- create an `AVCapturePhotoOutput` object
- create an `AVCapturePhotoSettings` object
- call the method `capturePhoto` on your `AVCapturePhotoOutput` object

```
func capturePhoto(settings: AVCapturePhotoSettings,
          delegate: AVCapturePhotoCaptureDelegate)
```

- this method will use the settings in capturing a photo, and begin calling methods on the delegate passed in (next slide lists some of these delegate methods)

AVCapturePhotoCaptureDelegate - some methods

`AVCapturePhotoCaptureDelegate` - monitors photo capture progress, including when the photo was finished processing.

```
// photo was taken - do with it what you will
optional func photoOutput(_ output: AVCapturePhotoOutput,
    didFinishProcessingPhoto photo: AVCapturePhoto,
    error: Error?)

// required for capturing live photos
optional func photoOutput(_ output: AVCapturePhotoOutput,
didFinishProcessingLivePhotoToMovieFileAt outputFileURL: URL,
                  duration: CMTime,
          photoDisplayTime: CMTime,
          resolvedSettings: AVCaptureResolvedPhotoSettings,
                     error: Error?)
```

AVCapturePhotoCaptureDelegate - some methods

```
// photo was taken - do with it what you will
optional func photoOutput(_ output:
AVCapturePhotoOutput,
    didFinishProcessingPhoto photo: AVCapturePhoto,
    error: Error?)
```

To get the Image “out of” the AVCapturePhoto object, you can use the following **AVCapturePhoto** method

```
// converts AVCapturePhoto object to a Data object
func fileDataRepresentation() -> Data?
```

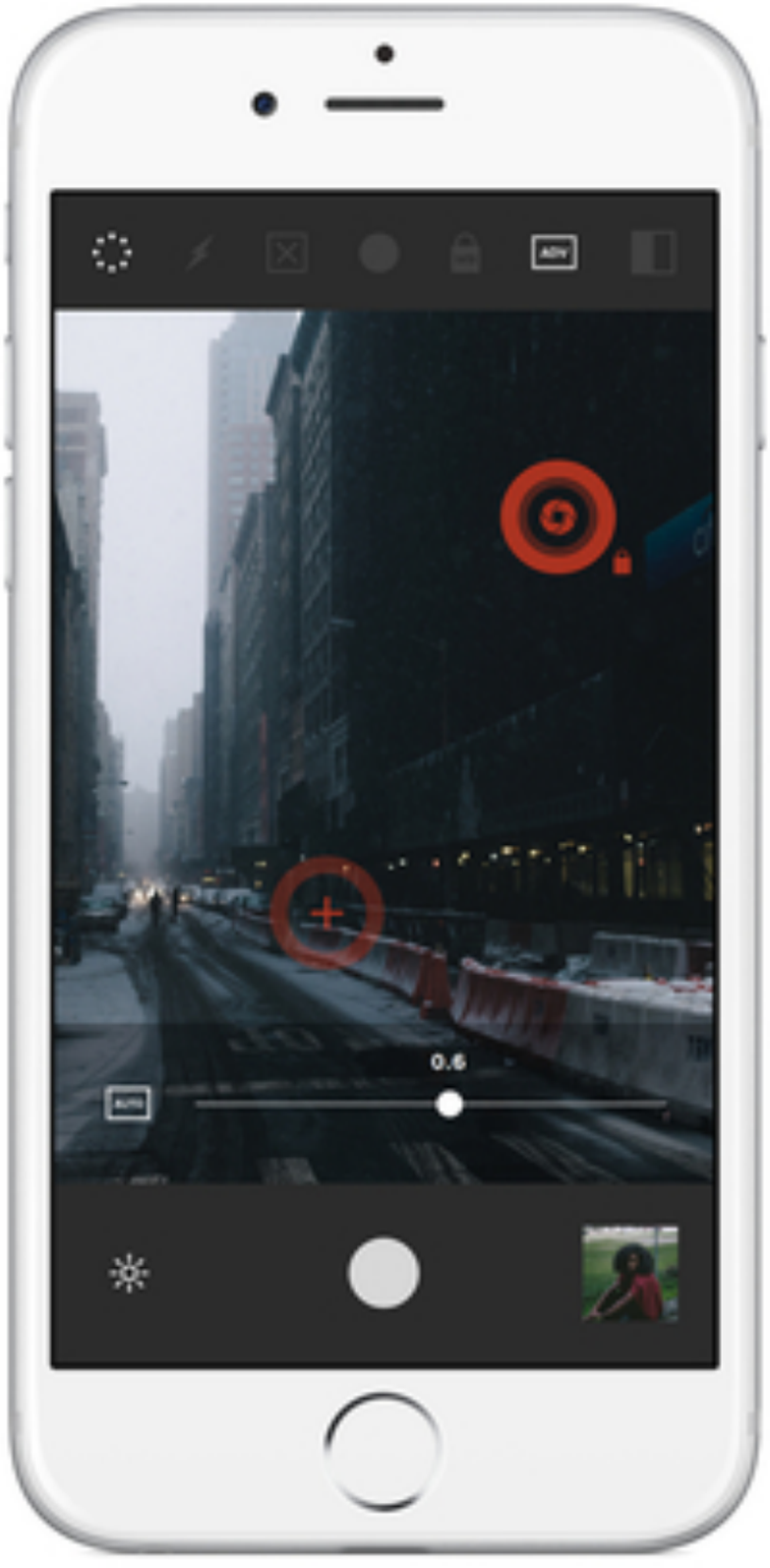
Check-in: tinyurl.com/ioscheckin

```
class PhotoViewController: UIViewController {
    // view controller implementation hidden
    let data: Data?

     @IBAction func takePhoto(_ sender: UIButton) {
         let photoSettings = AVCapturePhotoSettings()
         someClass.capturePhoto(with: photoSettings,
           delegate: self)
    }

    func photoOutput(_ output: AVCapturePhotoOutput,
                     didFinishProcessingPhoto photo:
                   AVCapturePhoto, error: Error?) {
        data = photo.fileDataRepresentation()
}}
```

Review

to capture AV data….

- Create a **AVCaptureSession**
- Set the **AVCaptureDeviceInput** depending on what you want to capture (video, photo, audio). Use AVCaptureDevice's DiscoverySession to get available inputs
- add outputs save data
- Begin media capture by calling `startRunning()` on your session
- use outputs to capture data from the session



VSCO