MAP KIT

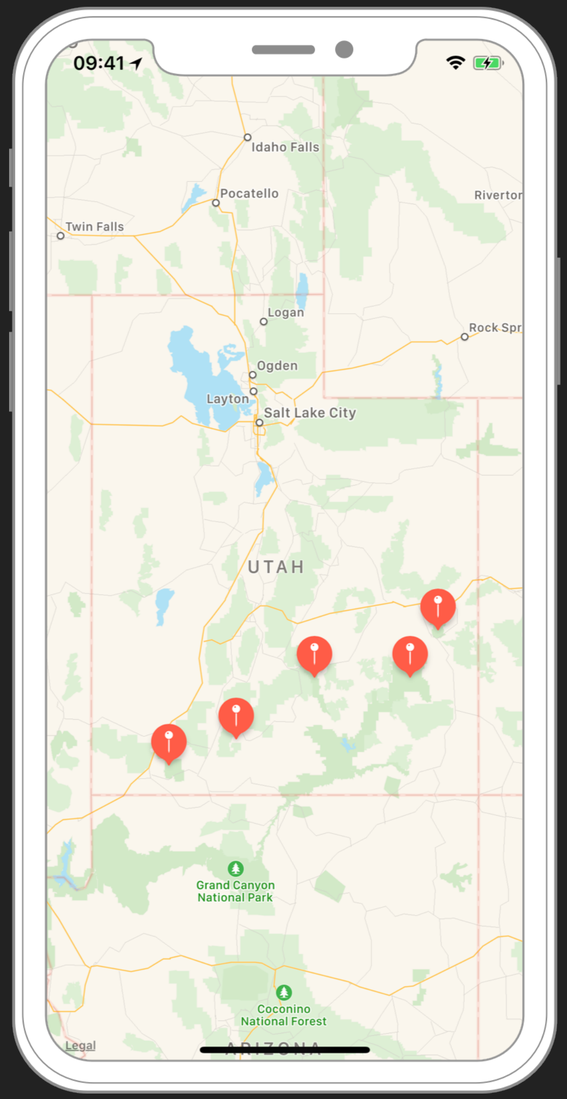
WHAT WE'LL COVER

- Annotations
- Annotation Views - specifically `MKMarkerAnnotationView`
- Overlays
- User Location
- Directions

ANNOTATIONS

ANNOTATIONS

- Defined by a single coordinate point
- Stay in place
- Annotation object
  - Conforms to `MKAnnotation` protocol
  - Manages the data for the annotation
- Annotation View
  - Used to draw the visual representation of the annotation on the map surface



DEFINE ANNOTATION OBJECT

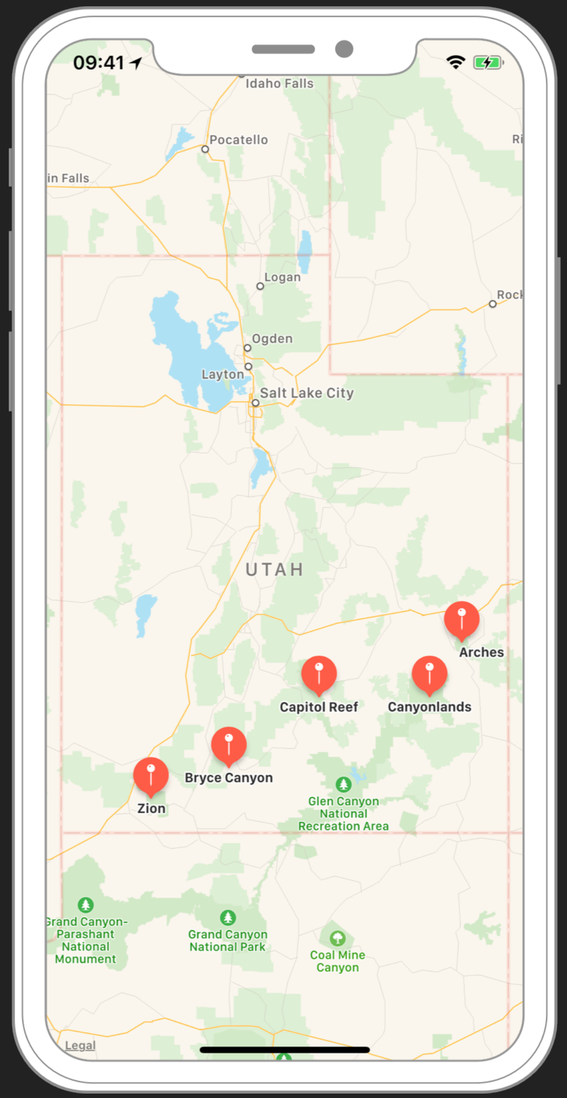
```
class Annotation: NSObject, MKAnnotation {

    //required
    var coordinate = CLLocationCoordinate2D()

    //optional
    var title: String?
    var subtitle: String?

    //your custom defined properties
    ...
}
```

DEFINE ANNOTATION OBJECT

```
import Foundation
import MapKit

class NationalPark: NSObject, MKAnnotation {

    //National Park Properties
    var name: String

    //National Park Methods

    init(name: String, coordinate: CLLocationCoordinate2D) {
        self.name = name
        self.coordinate = coordinate
        super.init()
    }

    // MARK: - MKAnnotation Protocol
    var coordinate: CLLocationCoordinate2D
    var title: String? {
        get { return name }
    }
}
```

ADD ANNOTATIONS

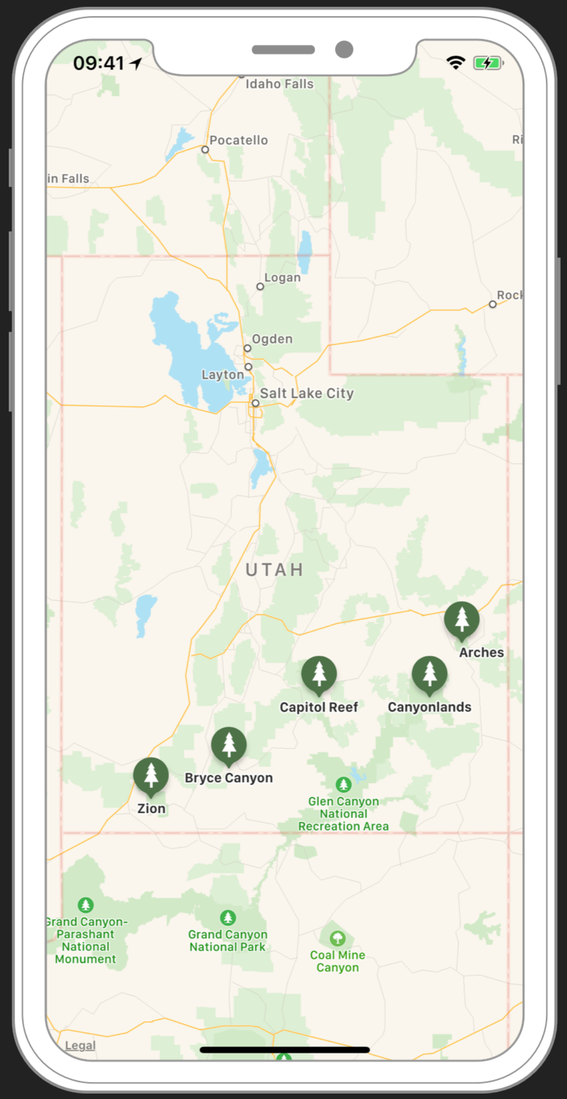
```
import UIKit
import MapKit

class ViewController: UIViewController, MKMapViewDelegate, CLLocationManagerDelegate {

    // MARK: - PROPERTIES
    @IBOutlet weak var mapView: MKMapView!

    func utahNationalParks() -> [NationalPark] {...}
    func utahCenterCoordinate() -> CLLocationCoordinate2D {...}

    override func viewDidLoad() {
        super.viewDidLoad()
        let utahRegion = MKCoordinateRegion(center: utahCenterCoordinate(),
                                            span: MKCoordinateSpan(latitudeDelta: 6.5,
                                                                   longitudeDelta: 6.5))

        mapView.region = utahRegion

        for nationalPark in utahNationalParks() {
            mapView.addAnnotation(nationalPark)
        }
    }
}
```


09:41
Idaho Falls
Pocatello
Logan
Ogden
Layton
Salt Lake City
UTAH
Arches
Capitol Reef
Canyonlands
Bryce Canyon
Zion
Glen Canyon National Recreation Area
Grand Canyon National Park
Coal Mine Canyon
Legal

ANNOTATION VIEWS

MARKER ANNOTATION VIEW

```
class ViewController: UIViewController, MKMapViewDelegate {
   ...
   func mapView(_ mapView: MKMapView, viewFor annotation: MKAnnotation) -> MKAnnotationView? {
      if annotation is NationalPark {
         var annotationView = mapView.dequeueReusableAnnotationView(withIdentifier: "Marker")
                              as? MKMarkerAnnotationView

         if let dequeuedAnnotationView = annotationView {
            dequeuedAnnotationView.annotation = annotation
         } else {
            annotationView = MKMarkerAnnotationView(annotation: annotation, reuseIdentifier: “Marker")
         }

         annotationView?.glyphImage = UIImage(named: “Tree")
         annotationView?.markerTintColor = UIColor(red: 78.0/255.0,
                                                   green: 114.0/255.0,
                                                    blue: 72.0/255.0,
                                                   alpha: 1.0)
         return annotationView
      }
      return nil
   }
}
```


09:41
Idaho Falls
Pocatello
Logan
Ogden
Layton
Salt Lake City
UTAH
Arches
Capitol Reef
Canyonlands
Bryce Canyon
Zion
Glen Canyon National Recreation Area
Grand Canyon National Park
Coal Mine Canyon
Legal

OVERLAYS

OVERLAYS

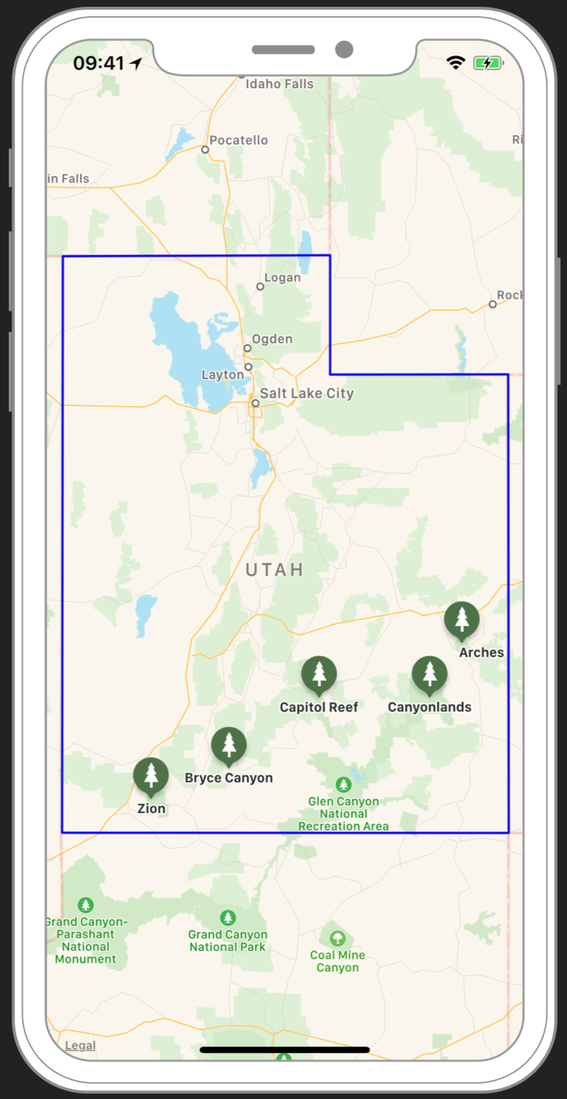
- Content that is defined by any number of points
  - May constitute one or more, contiguous or noncontiguous shape(s)
- Pre-defined overlays:
  - `MKCircle`
  - `MKPolyline`
  - `MKPolygon`

OVERLAY RENDERS

- Need an `MKOverlayRenderer` object:
  - `MKCircleRenderer`
  - `MKPolylineRenderer`
  - `MKPolygonRenderer`
- The renderer object defines the overlay style:
  - Fill color
  - Line color
  - Line width
  - etc…
- Specify your renderer object in the map view delegate
  - `mapView(_:rendererFor:)`

OVERLAY

```
class ViewController: UIViewController, MKMapViewDelegate {

   @IBOutlet weak var mapView: MKMapView!

   func utahCoordinates() -> [CLLocationCoordinate2D] {…}

   override func viewDidLoad() {
      ...
      let utahPolygon = MKPolygon(coordinates: utahCoordinates(), count: utahCoordinates().count)
      mapView.add(utahPolygon)
   }

   func mapView(_ mapView: MKMapView, rendererFor overlay: MKOverlay) -> MKOverlayRenderer {
      if overlay is MKPolygon {
         let renderer = MKPolygonRenderer(overlay: overlay)
         renderer.lineWidth = 1.5
         renderer.strokeColor = .blue
         return renderer
      } else {
         return MKOverlayRenderer()
      }
   }
}
```


09:41
Idaho Falls
Pocatello
Logan
Ogden
Layton
Salt Lake City
UTAH
Arches
Capitol Reef
Canyonlands
Bryce Canyon
Zion
Glen Canyon National Recreation Area
Grand Canyon-Parashant National Monument
Grand Canyon National Park
Coal Mine Canyon
Legal

USER LOCATION

USER LOCATION SERVICES

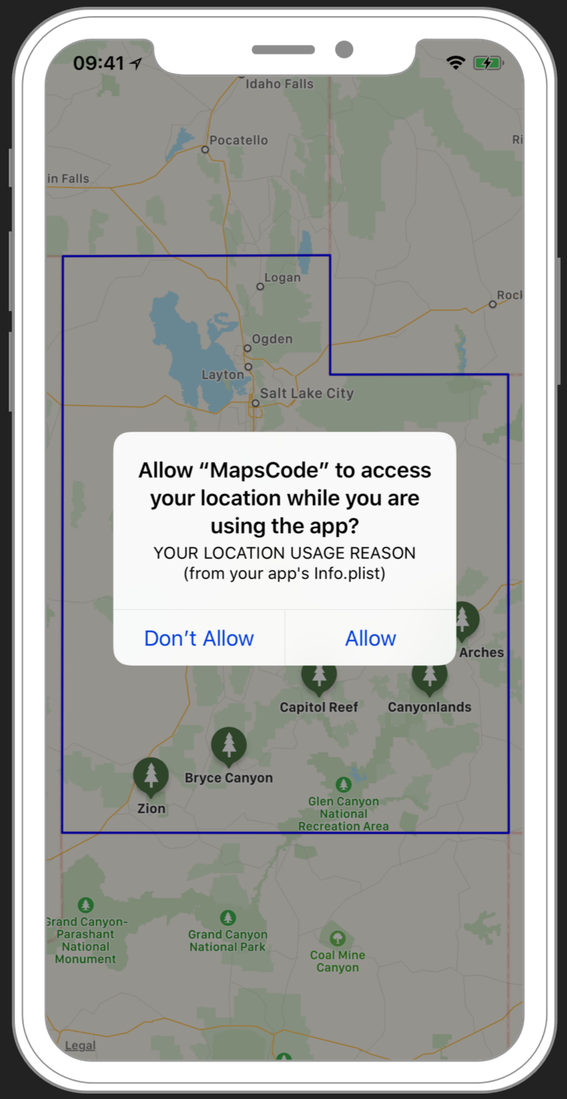
- Core Location framework
  - Standard location service
  - Significant-change location service
- Power-intensive operation

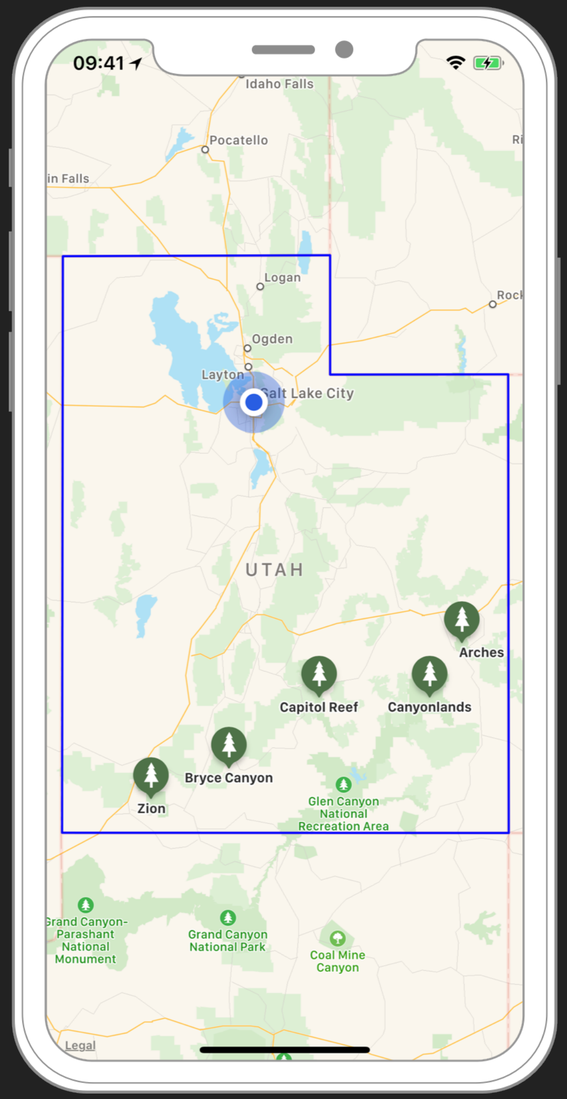
LOCATION MANAGER

- Use `CLLocationManager` class to start and stop location services
- Use `requestWhenInUseAuthorization()` or `requestAlwaysAuthorization()` to ask for user permission
- Use Privacy - Location When In Use Usage Description or Privacy - Location Always Usage Description keys in info.plist to describe how you use the user's location
- Use the `CLLocationManagerDelegate` to monitor location events
- Before starting location services, Apple recommends you call `CLLocationManager.locationServicesEnabled()`

STANDARD LOCATION SERVICES

```
class ViewController: UIViewController, MKMapViewDelegate, CLLocationManagerDelegate {
   @IBOutlet weak var mapView: MKMapView!
   var locationManager = CLLocationManager()

   override func viewDidAppear(_ animated: Bool) {
      super.viewDidAppear(animated)

      askForLocationPermissions()
   }

   func askForLocationPermissions() {
      locationManager.requestWhenInUseAuthorization()
      locationManager.delegate = self
      locationManager.desiredAccuracy = kCLLocationAccuracyBest

      if CLLocationManager.locationServicesEnabled() {
         locationManager.startUpdatingLocation()
         mapView.showsUserLocation = true
      }
   }

   // MARK: - LOCATION MANAGER DELEGATE
   func locationManager(_ manager: CLLocationManager, didChangeAuthorization status: CLAuthorizationStatus) {
      if status == .authorizedWhenInUse {
         manager.startUpdatingLocation()
         mapView.showsUserLocation = true
      }
   }
}
```


09:41
Idaho Falls
Pocatello
Logan
Ogden
Layton
Salt Lake City
Allow “MapsCode” to access your location while you are using the app?
YOUR LOCATION USAGE REASON
(from your app's Info.plist)
Don’t Allow
Allow
Arches
Capitol Reef
Canyonlands
Bryce Canyon
Zion
Glen Canyon National Recreation Area
Grand Canyon-Parashant National Monument
Grand Canyon National Park
Coal Mine Canyon
Legal


09:41
Idaho Falls
Pocatello
Logan
Ogden
Layton
Salt Lake City
UTAH
Arches
Capitol Reef
Canyonlands
Bryce Canyon
Zion
Glen Canyon National Recreation Area
Grand Canyon-Parashant National Monument
Grand Canyon National Park
Coal Mine Canyon
Legal

DIRECTIONS

GENERAL PURPOSE DIRECTION INFORMATION

- Use the `MKDirections` API to get information about a route:
  - Distance
  - Expected travel time
  - Localized advisory notices
  - Individual steps that make up the route
- Server-based and require a network connection

STEPS TO GET DIRECTION INFORMATION

1. Create an `MKDirectionsRequest` object and configure it with start and end `MKMapItem` objects

2. Create an `MKDirections` object and initialize it with the `MKDirectionsRequest` object you created in step 1.

3. Call `calculateDirectionsWithCompletionHandler:` to start the route finding process

4. Handle the route information contained in the `MKDirectionsResponse` object

DIRECTION CODE

```
class ViewController: UIViewController, MKMapViewDelegate, CLLocationManagerDelegate {

   ...

   func getDrivingDirectionsFromCurrentLocation() {
      //1
      let drivingRouteRequest = MKDirectionsRequest()
      drivingRouteRequest.transportType = .automobile
      drivingRouteRequest.source = MKMapItem(placemark: MKPlacemark(coordinate: mapView.userLocation.coordinate))
      let zion = utahNationalParks()[0]
      drivingRouteRequest.destination = MKMapItem(placemark: MKPlacemark(coordinate: zion.coordinate))
      //2
      let drivingRouteDirections = MKDirections(request: drivingRouteRequest)
      UIApplication.shared.isNetworkActivityIndicatorVisible = true
      //3
      drivingRouteDirections.calculate { (response: MKDirectionsResponse?, error: Error?) in
         DispatchQueue.main.async { UIApplication.shared.isNetworkActivityIndicatorVisible = false }
         //4
         guard let response = response else { print("\(String(describing: error?.localizedDescription))”); return }

         guard let firstRoute = response.routes.first else { print("No Routes”); return }

         if let routePolyline = self.routePolyline {
            self.mapView.remove(routePolyline)
         }

         self.routePolyline = firstRoute.polyline

         DispatchQueue.main.async { self.mapView.add(firstRoute.polyline) }
       }
    }
}
```

DIRECTION CODE – STEP 1

```
let drivingRouteRequest = MKDirectionsRequest()
drivingRouteRequest.transportType = .automobile

let sourcePlacemark = MKPlacemark(coordinate: mapView.userLocation.coordinate)
drivingRouteRequest.source = MKMapItem(placemark: sourcePlacemark)

let zion = utahNationalParks()[0]
let destinationPlacemark = MKPlacemark(coordinate: zion.coordinate)
drivingRouteRequest.destination = MKMapItem(placemark: destinationPlacemark)
```

DIRECTION CODE – STEP 2

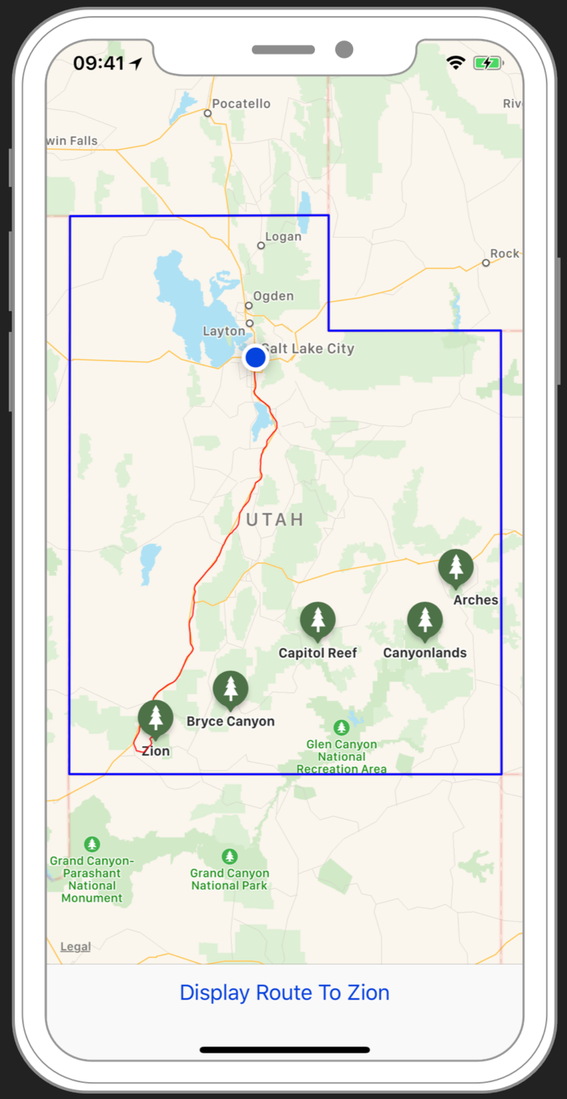
```
let drivingRouteDirections = MKDirections(request: drivingRouteRequest)
UIApplication.shared.isNetworkActivityIndicatorVisible = true
```

DIRECTION CODE – STEP 3

```
drivingRouteDirections.calculate { (response: MKDirectionsResponse?, error: Error?) in
    DispatchQueue.main.async {
        UIApplication.shared.isNetworkActivityIndicatorVisible = false
    }









 }
```

DIRECTION CODE – STEP 4

```
drivingRouteDirections.calculate { (response: MKDirectionsResponse?, error: Error?) in
    DispatchQueue.main.async {
        UIApplication.shared.isNetworkActivityIndicatorVisible = false
    }

    guard let response = response else {
        print("\(String(describing: error?.localizedDescription))"); return
    }

    guard let firstRoute = response.routes.first else { print("No Routes"); return }

    if let routePolyline = self.routePolyline {
        self.mapView.remove(routePolyline)
    }

    self.routePolyline = firstRoute.polyline

    DispatchQueue.main.async { self.mapView.add(firstRoute.polyline) }
 }
```


09:41
Pocatello
Logan
Ogden
Layton
Salt Lake City
UTAH
Arches
Capitol Reef
Canyonlands
Bryce Canyon
Zion
Glen Canyon National Recreation Area
Grand Canyon-Parashant National Monument
Grand Canyon National Park
Legal
Display Route To Zion

QUESTIONS?

WORKS CITED

- Location and Maps Programming Guide
  - https://developer.apple.com/library/content/documentation/UserExperience/Conceptual/LocationAwarenessPG/Introduction/Introduction.html
- “What’s New in MapKit”
  - https://developer.apple.com/videos/play/wwdc2017/237/