MapKit

Mapkit is used to show a mapview in your app. The framework is also responsible for annotating and displaying pins on the map, it can also be used to for get the coordinates of an address to make a placemark, or from a coordinate to an address (reverseGeocoding).

Before starting:

In order to use a map and find out the users location you need to import the MapKit.framework to your project. If you don't know how to do that, then follow the steps bellow to add any library or framework to your project.

Adding a library or a framework to your project:

1. Click on your project and go to the General menu.
2. Open the Link Binary With Libraries section, click the Plus button, find or search for the entry for MapKit.framework, and click Add.
3. To add any other Library of framework, just replace Mapkit.framework from step number two with the name of the framework or library.

Setting up the Map:

StoryBoard:

To set up a map you'll have to add a Map View to your desired view controller, that is about all you have to do to set up a map in the storyboard.

1. Drag a Map View to your view controller.
2. Check "Shows user location" in the attributes tab, if you want the map to display the users current location.

If you run the project now, the map will be displayed, but you might not be able to see your current location (Since you're using the simulator). If you want to check if it's displaying the users location, then you can click on the debug menu in the simulator and click on location, you can choose one of the locations available in the menu, or you can specify your own location by clicking custom location within the menu and supply it with a latitude and longitude.

At this point, your project should show the world map and a pin indicating your current(simulated) location.

The ViewController:

You have the map setup in the storyboard, but there is nothing that tells your ViewController that it has a map. You need to link the Map View to your ViewController in order to have the map do all sorts of interesting things.

Open the assistant editor(butler icon) and set it to viewController.h. Then control drag from the map view in the storyboard to viewContoller.h below the @implementation. You might notice that it gives you a bunch of errors, and that is because we haven't imported mapKit, just add the following to the .h file:

```
#import <MapKit/MapKit.h>
```

Finally, add the following piece of code to your viewDidLoad() method `self.mapView.delegate = self;`. And, that's it for setting up a map and linking it to a view controller.

Zooming & Visible Area:

Zooming into the user's location:

If you want to zoom into an area that is 800x800 meters around the users location add the following method to your viewController.m. This is what we want to do in this project.

```
- (void)mapView:(MKMapView *)mapView didUpdateUserLocation:(MKUserLocation
*)userLocation
{
    MKCoordinateRegion region =
MKCoordinateRegionMakeWithDistance(userLocation.coordinate, 800, 800);
    [self.mapView setRegion:[self.mapView regionThatFits:region] animated:YES];
}
```

Zooming into any location & Setting up viewable region:

Here's a general template to set the viewable region and zoom into a location:

```
- (void)mapView:(MKMapView *)mapView didUpdateUserLocation:(MKUserLocation
*)userLocation
{
    //Set the coordinates of the location
    //Change lat and long with the location's latitude and longitude
    CLLocationCoordinate2D zoomLocation;
    zoomLocation.latitude = lat;
    zoomLocation.longitude= long;


    int zoomRegion = 500;//Change with custom region

    MKCoordinateRegion region = MKCoordinateRegionMakeWithDistance(zoomLocation,
                                region, region);

    [self.mapView setRegion:[self.mapView regionThatFits:region] animated:YES];
}
```

Creating and Adding Annotations:

Displaying a pin with information on the map is easy. You'll need to create an MKPointAnnotation with a coordinate, a title, and a subtitle for the annotation you want to display for the location. Which can be done in a manner similar to the snippet:

```
//Set the coordinates of the location
    //Change lat and long with the location's latitude and longitude
    CLLocationCoordinate2D locationCoordinate;
    location.latitude = lat;
    location.longitude= long;

    // Create the annotation
    MKPointAnnotation *point = [[MKPointAnnotation alloc] init];
    point.coordinate = locationCoordinate;
    point.title = @"Location Name";
    point.subtitle = @"Address1";

    //Add the annotation to the map
    [self.mapView addAnnotation:point];
```

These annotations can be added to the map in several way. For example, you can add the MKPointAnnotation to your map before it displays by placing it in `viewDidLoad`. Or, when the user moves, (location has been updated) by placing it inside – `(void)mapView:(MKMapView *)mapView didUpdateUserLocation:(MKUserLocation *)userLocation`.

GeoCoding

Apple provides us with a neat class called CLGeocoder which enables us to convert to and from a coordinate(longitude and latitude) and an address. There are no rigid restriction for the address form. Think of google maps, let's say you typed in "700 bay state road" the autocomplete will show you all the different "700 bay state road"'s out there (Boston, Cambridge, Quincy, and Belmont). That's exactly what you are going to get if you provide it as an address to the CLGeocoder; an array of locations (of type CLPlacemark).

You'll need to link the `CoreLocation.framework` and import the following to the .h file wherever you are using geocoding `#import <CoreLocation/CoreLocation.h>`.

Finding the Coordinates of an Address:

It's a very common problem where you know an address and you want to add it as an annotation to the map. Now the mapView can only let you annotate coordinates or points, so you will most likely have to find the coordinates of the address

Example:

```
//The address that we want to find coordinates for
    NSString *address = @"theAddress";

    //Create the coder
    CLGeocoder *geocoder = [[CLGeocoder alloc] init];


    [geocoder geocodeAddressString: address completionHandler:^(NSArray* placemarks,
NSError* error){

        //All the results are contained in placemarks

        //Start processing each placemark
        for (CLPlacemark* aPlacemark in placemarks)
        {
            // Process the placemark.

            //Getting the coordinates from the placemark
            CLLocationCoordinate2D addressCoordinates;

            addressCoordinates.latitude = aPlacemark.location.coordinate.latitude;
            addressCoordinates.longitude = aPlacemark.location.coordinate.longitude;

        }



        //Call method depending on data here
        //(Look at Word of Caution)
    }];
```

Getting an Address from a Coordinate:

To convert a set of coordinates to its corresponding address, we use the reverseGeocodeLocation method which takes in a CLLocation. A CLLocation is an object that contains a CLLocationCoordinate2D struct the contains the latitude and longitude. The result will usually be one item.

The address stores in the placemark will be in a for of dictionary, with each unique aspect of an address is a key that can be used to acces the value. Look at the example below.

Example:

```
CLLocation *location = [[CLLocation alloc] init];

    location = [[CLLocation alloc] initWithLatitude: latitude longitude:longitude];

    latitude = lat;
    longitude= longit;

    [geocoder reverseGeocodeLocation:location completionHandler:
     ^(NSArray* placemarks, NSError* error){

         /*
          *No need for  a for loop because coordinates are usually associated with
one address.
          *So, we only need to check if it has found an address with the specified
coordinate
          */

         if ([placemarks count] > 0)
         {

             //Process parts of the address to a string

             CLPlacemark *mark = [placemarks objectAtIndex:0];

             NSDictionary *addressDict = mark.addressDictionary;

             NSArray *keys = [[NSArray alloc] initWithObjects:@"Street", @
                                  "City", @"State",@ "ZIP", nil];
             NSString *address;
             NSString *temp;

             for (NSString *key in keys) {

                 temp = [addressDict objectForKey:key];

                 if (temp)
                 {
                     address = [NSString stringWithFormat:@" %@", temp];
                 }

             }

             NSLog(@"The address is: %@",address );

         }
     }];
```

Word of Caution:

The geocoder runs asynchronously(a different thread/ flow of execution), wich means if your app depends immediately on the results of the geocoder, it might seem as if you ended up with incorrect or no data from the process. This is because the geocoder method enters a block which creates a new thread with its own workflow and process, and then returns the results of the process. The method you have the block in will still run and it won't wait for the actual geocoder process result to return.

A simple way to circumvent the consequences of this asynchronus process, is to call the method that depends on the data at the end of the block.

Resources:

- Locations and Maps Programming Guide
- A Demo Project
- A past GAI project, (Peer Health Exchange) that makes great use of mapKit.
- MKDirection tutorials:

- DevFright: MKDirection Tutorial
- Shinobi Controls: Route Directions with Mapkit