Core Location & Mapkit

IRON
HACK

Leo Font

Core Location

- Include CoreLocation Framework
- import <CoreLocation/CoreLocation.h>
- CLLocationManager entry point class for location services
- You have to conform to <CLLocationManagerDelegate> to receive updates

```
-(void)locationManager:(CLLocationManager *)manager didUpdateLocations:(NSArray *)locations

- (void)locationManager:(CLLocationManager *)manager didUpdateHeading:(CLHeading *)newHeading

- (BOOL)locationManagerShouldDisplayHeadingCalibration:(CLLocationManager *)manager
```

Ask for position

- Check if services available (turned on/off):

```
if ([CLLocationManager locationServicesEnabled])
if ([CLLocationManager headingAvailable])
```

- New iOS8: Ask for permission:

```
if ([self.locationManager respondsToSelector:
    @selector(requestWhenInUseAuthorization)])
{
        [self.locationManager requestWhenInUseAuthorization];
}
```

- Add to plist: NSLocationWhenInUseUsageDescription

Set parameters and start updating

```
[self.locationManager setDesiredAccuracy:kCLLocationAccuracyBest];
    [self.locationManager startUpdatingLocation];
    [self.locationManager startUpdatingHeading];
```


KEEP
CALM
IT'S
YOUR
TURN NOW

Mapkit

- Include MapKit Framework
- Import <<MapKit/MapKit.h>
- Add a MKMapView to the View
- Conform to <MKMapViewDelegate>

Mapkit

- Properties (you can set them in IB or code):
  - zoomEnabled
  - scrollEnabled
  - pitchEnabled (3d perspective)
  - rotateEnabled
  - showsUserLocation
  - showsBuildings / - showsPointsOfInterest

Mapkit

- Map informs about position change with delegate calls:

```
-(void)mapView:(MKMapView *)mapView didUpdateUserLocation:(MKUserLocation *)userLocation
```

Mapkit

- We can center map position:

```
 // Hand made coords

CLLocationCoordinate2D madrid  = CLLocationCoordinate2DMake(40.4167,-3.7037);

// Or use real user coords
CLLocationCoordinate2D currentUserCoord = self.map.userLocation.coordinate;


MKCoordinateRegion viewRegion =
MKCoordinateRegionMakeWithDistance(currentUserCoord, 1000, 1000);

MKCoordinateRegion adjustedRegion = [self.map regionThatFits:viewRegion];

[self.map setRegion:adjustedRegion animated:YES];

// Fires delegate call:
-(void)mapView:(MKMapView *)mapView regionDidChangeAnimated:(BOOL)animated
```

MapType

-You can set which type of map you want to show:

```
typedef enum : NSUInteger {
    MKMapTypeStandard,
    MKMapTypeSatellite,
    MKMapTypeHybrid
} MKMapType;
```

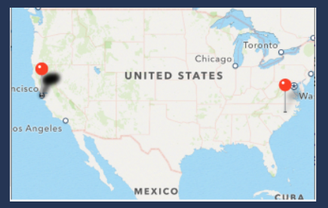

KEEP
CALM
IT'S
YOUR
TURN NOW

Annotations

- An annotation is any object that conforms to MKAnnotation
- Protocol **requires**

```
@property (nonatomic) CLLocationCoordinate2D coordinate;
```

- title, subtitle: optional propierties
- Default aspect = pin



Adding/Removing Annotations

```
// Adding

MyPinClass *pin = [[MyPinClass alloc]init];
pin.coordinate = coordinate;
[map addAnnotation: pin];

 // Removing


[self.map renoveAnnotations:[self.map annotations]];
```

Custom Annotations

```
- (MKAnnotationView *)mapView:(MKMapView *)mapView viewForAnnotation:(id<MKAnnotation>)annotation {
    MKAnnotationView *mkView;

// Delegates being called depending on Callout

-(void)mapView:(MKMapView *)mapView annotationView:(MKAnnotationView *)view calloutAccessoryControlTapped:(UIControl *)control

-(void)mapView:(MKMapView *)mapView didDeselectAnnotationView:(MKAnnotationView *)view
```


KEEP
CALM
IT'S
YOUR
TURN NOW

Local Search

- MKLocalSearch allows to find nearby points of interest within a geographic region.
- MKLocalSearchRequest takes a naturalLanguageQuery, such as “Restaurants”, and an optional bounding geographic region to constrain results
- It returns an array of MKMapItem objects

Local Search

```
MKLocalSearchRequest *request = [[MKLocalSearchRequest alloc] init];
request.naturalLanguageQuery = @"Restaurants";
request.region = mapView.region;
MKLocalSearch *search = [[MKLocalSearch alloc] initWithRequest:request];
[search startWithCompletionHandler:^(MKLocalSearchResponse *response, NSError *error) {
    if (!error){
      NSLog(@“Response!”;
      for (NSDictionary *poi in response.mapItems) {
        NSLog (@“%@“,poi);
          }
        }

}];
```


KEEP
CALM
IT'S
YOUR
TURN NOW

Geocoding

- CLGeocoder performs geocoding by sending it a String or Dictionary with an address.

```
- (void)geocodeAddressDictionary:(NSDictionary *)addressDictionary
                completionHandler:(CLGeocodeCompletionHandler)completionHandler
```

```objc
NSMutableDictionary *placeDictionary = [[NSMutableDictionary alloc] init];

    NSArray *keys = @[@"Street", @"City"];
    NSArray *addressComponents  = [address componentsSeparatedByString:@","];

    if (addressComponents.count == 2) {
        [addressComponents enumerateObjectsUsingBlock:^(id obj, NSUInteger idx, BOOL *stop) {
            [placeDictionary setObject:obj forKey:keys[idx]];
        }];
    }

    CLGeocoder *geocoder = [[CLGeocoder alloc] init];
    [geocoder geocodeAddressDictionary:placeDictionary completionHandler:^(NSArray *placemarks, NSError *error) {
        if([placemarks count]) {
            CLPlacemark *placemark = [placemarks objectAtIndex:0];

            CLLocation *location = placemark.location;
            CLLocationCoordinate2D coordinate = location.coordinate;

            [self centerMap:self.map InCoordinate:coordinate];
        } else {
            NSLog(@"error");
        }
    }];
```

Reverse Geocoding

- CLGeocoder can also perform reverse geocoding. Providing it with a latitude and longitude returns a dictionary of Placemarks.

```
- (void)reverseGeocodeLocation:(CLLocation *)location
                completionHandler:(CLGeocodeCompletionHandler)completionHandler
```

Reverse Geocoding

```
CLGeocoder *geocoder = [[CLGeocoder alloc] init];
  [geocoder reverseGeocodeLocation:self.selectedLocation
  completionHandler:^(NSArray *placemarks, NSError *error) {

      if(placemarks.count){
          NSDictionary *dictionary = [[placemarks objectAtIndex:0] addressDictionary];

      NSMutableString *s = [NSMutableString stringWithFormat:@"%@", [dictionary valueForKey:@"Street"]];
          [s appendString:[dictionary valueForKey:@"City"]];
          [s appendString:[dictionary valueForKey:@"State"]];
          [s appendString:[dictionary valueForKey:@"ZIP"]];

          self.textSearch.text = s;

          }
  }];
```


KEEP
CALM
IT'S
YOUR
TURN NOW

Maps 3D Camera

- A camera object defines a point above the map's surface from which to view the map.

  -Applying a camera to a map has the effect of giving the map a 3D-like appearance.

```
MKMapCamera *camera = [MKMapCamera cameraLookingAtCenterCoordinate:coordinate
                                                     fromEyeCoordinate:coordinate eyeAltitude:altitude];

[self.map setCamera:camera];
```


KEEP
CALM
IT'S
YOUR
TURN NOW

Calculating and presenting routes

- MKDirections generates directions from one geographical location to another.
- Multiple parameters can be specified to refine route alternatives or route for car or walking.

Generating request and Callback

```
MKDirectionsRequest *request = [[MKDirectionsRequest alloc] init];

    request.source = [MKMapItem mapItemForCurrentLocation];
    request.transportType = MKDirectionsTransportTypeWalking;

    MKPlacemark *destinationPlacemark = [[MKPlacemark alloc] initWithCoordinate:self.selectedLocation.coordinate
                                                  addressDictionary:nil] ;

    MKMapItem *destinationMapItem = [[MKMapItem alloc] initWithPlacemark:destinationPlacemark];


    request.destination = destinationMapItem;
    request.requestsAlternateRoutes = YES;
    MKDirections *directions =
    [[MKDirections alloc] initWithRequest:request];

    [directions calculateDirectionsWithCompletionHandler:
     ^(MKDirectionsResponse *response, NSError *error) {
         if (error) {
             // Handle Error
         } else {
             [self showRoute:response];
         }

     }];
```

Drawing map overlay & rendering

```
 for (MKRoute *route in response.routes)
    {
        [self.map
         addOverlay:route.polyline level:MKOverlayLevelAboveRoads];

        for (MKRouteStep *step in route.steps)
        {
            NSLog(@"%@", step.instructions);
       // Step by step instructions can be processed (voice synthesis, display)
        }
    }

  // rendering via delegate Map method

- (MKOverlayRenderer *)mapView:(MKMapView *)mapView rendererForOverlay:(id < MKOverlay >)overlay
{
    MKPolylineRenderer *renderer =
    [[MKPolylineRenderer alloc] initWithOverlay:overlay];
    renderer.strokeColor = [UIColor blueColor];
    renderer.lineWidth = 5.0;
    return renderer;
}
```


KEEP
CALM
IT'S
YOUR
TURN NOW