Blocks

- Demo block

```
@interface C : NSObject { int i; }
@property int i;
@end

@implementation C
@synthesize i;
@end

int main (int argc, const char * argv[]) {
    NSAutoreleasePool* pool = [[NSAutoreleasePool alloc] init];

    C* c = [[C alloc] init];
    NSLog(@"value of %@ is %d", c, c.i);

    typedef void (^F)();
    F f = ^ {
         c.i = 42;                              // [c retain]; !!!
         NSLog(@"value of %@ is %d", c, c.i);
    }; // in main's scope only

    NSDictionary *dict =
    [NSDictionary dictionaryWithObject: [[f copy] autorelease]forKey: @"doIt"];

    [c release]; // NOT GONE, BECAUSE retained by f's copy

    // ([dict objectForKey:@"doIt"])(); // NOT DIRECTLY CALLABLE
    void (^call)() = [dict objectForKey:@"doIt"];
    call();

    [pool drain]; return 0;
}
```

Blocks

- Wozu werden Blöcke in iOS benutzt?
  - Enumerations
  - View Animations
  - Sorting (sortieren mit einem Block als comparison method)
  - Notification (wenn irgend etwas passiert, führe diesen Block aus)
  - Error handlers (bei einem Fehler, führe diesen Block aus)
  - Completion handlers (nachdem etwas beendet ist, führe diesen Block aus)
  - und am wichtigsten: Multithreading mit dem Grand Central Dispatch API

Grand Central Dispatch GCD

- ein C-API (libdispatch: open source)
- Grundidee: separate Ausführungs-Queues, in die man Blöcke per FIFO einstellen kann
  - diese werden pro Queue (zumeist) der Reihe nach ausgeführt
  - verschiedene Queues werden unabhängig voneinander in separaten Threads abgearbeitet
  - blockiert die Abarbeitung einer Queue, sind andere davon nicht betroffen
  - Hauptanwendung: Verlagerung blockierender Aktionen (Netzzugriff o.ä.) aus dem user-interface (main) thread

Grand Central Dispatch

- die wichtigsten Funktionen:

```
// Creating and releasing queues
dispatch_queue_t dispatch_queue_create(const char *label, NULL);
void dispatch_release(dispatch_queue_t);

// Putting blocks in the queue
typedef void (^dispatch_block_t)(void);
void dispatch_async(dispatch_queue_t queue, dispatch_block_t block);

// Getting the current or main queue
dispatch_queue_t dispatch_get_current_queue();
dispatch_queue_t dispatch_get_main_queue();

// and much much more ....
```

Grand Central Dispatch

- Demo blocks

```
#include <stdio.h>
#include <dispatch/dispatch.h>

int main(int argc, char *argv[])
{
   void (^block)(int, int);
   block=^(int n, int c){for (int i=0; i<n; ++i)
      putchar(c);
   };

   dispatch_queue_t q1=dispatch_queue_create("q1", NULL);
   dispatch_queue_t q2=dispatch_queue_create("q2", NULL);
   dispatch_async(q1, ^{block(1000,'A');});
   dispatch_async(q2, ^{block(1000,'B');});
   block(1000, 'C');
}
```

Grand Central Dispatch

- Demo ticktack

```
#include <stdio.h>
#include <time.h>
#include <stdlib.h>
#include <dispatch/dispatch.h>

int main (int argc, const char * argv[]) {
    __block int n=0;
    dispatch_queue_t q_default;
    q_default = dispatch_get_global_queue(DISPATCH_QUEUE_PRIORITY_DEFAULT, 0);
    dispatch_source_t timer =
        dispatch_source_create(DISPATCH_SOURCE_TYPE_TIMER, 0, 0, q_default);
        //run event handler on the default global queue
    dispatch_time_t now = dispatch_walltime(DISPATCH_TIME_NOW, 0);
    dispatch_source_set_timer(timer, now, 1ull*NSEC_PER_SEC, 5000ull);
    dispatch_source_set_event_handler(timer, ^{
                                          printf("%s\n", (++n%2)?"tick":"tack");
                                          });
    dispatch_resume(timer);
   for(;;){}
   return 0;
}
```

Grand Central Dispatch

- Beispiel

```
- (void)viewWillAppear:(BOOL)animated
{
    NSData *imageData = [WebFetcher imageDataForImageWithURLString:
                                                    someImage.URL];
    UIImage *image = [UIImage imageWithData:imageData];
    self.imageView.image = image;
    self.imageView.frame = CGRectMake
                           (0,0,image.size.width,image.size.height);
    self.scrollView.contentSize = image.size;
}
```

blockiert UI

Grand Central Dispatch

- Beispiel: besser, aber falsch

```
- (void)viewWillAppear:(BOOL)animated
{
    dispatch_queue_t downloadQueue =
                    dispatch_queue_create(“Downloader”, NULL);
    dispatch_async(downloadQueue,
    ^{
          NSData *imageData =
            [WebFetcher imageDataForImageWithURLString:someImage.URL];
          UIImage *image = [UIImage imageWithData:imageData];
          self.imageView.image = image;
          self.imageView.frame = CGRectMake(/*as before*/...);
          self.scrollView.contentSize = image.size;
    });
}
```

UIKit Aufrufe nur im main-Thread erlaubt :-(

Grand Central Dispatch

- Beispiel: besser, fast richtig

nicht Thread-safe, falls NSManagedContext @dynamic :-(

```
- (void)viewWillAppear:(BOOL)animated {
    dispatch_queue_t downloadQueue =
        = dispatch_queue_create("Downloader", NULL);
    dispatch_async(downloadQueue,
    ^{  NSData *imageData =
            [WebFetcher imageDataForImageWithURLString:someImage.URL];
        dispatch_async(dispatch_get_main_queue(),
        ^{
            UIImage *image = [UIImage imageWithData:imageData];
            self.imageView.image = image;
            self.imageView.frame = CGRectMake(...);
            self.scrollView.contentSize = image.size;
        });
    });
}
```

Grand Central Dispatch

- **Beispiel:** 99% korrekt (NSManagedObject-Zugriff im main-Thread)

```
- (void)viewWillAppear:(BOOL)animated {
    NSString *url = someImage.URL;
    dispatch_queue_t downloadQueue =
        = dispatch_queue_create("Downloader", NULL);
    dispatch_async(downloadQueue,
    ^{  NSData *imageData =
            [WebFetcher imageDataForImageWithURLString:url];
        dispatch_async(dispatch_get_main_queue(),
        ^{
            UIImage *image = [UIImage imageWithData:imageData];
            self.imageView.image = image;
            self.imageView.frame = CGRectMake(...);
            self.scrollView.contentSize = image.size;
        });
    });
}
```

Grand Central Dispatch

- **Beispiel:** 100% korrekt (kein Queue-Leak)

```
- (void)viewWillAppear:(BOOL)animated {
    NSString *url = someImage.URL;
    dispatch_queue_t downloadQueue =
        = dispatch_queue_create("Downloader", NULL);
    dispatch_async(downloadQueue,
    ^{  NSData *imageData =
            [WebFetcher imageDataForImageWithURLString:url];
        dispatch_async(dispatch_get_main_queue(),
        ^{
            UIImage *image = [UIImage imageWithData:imageData];
            self.imageView.image = image;
            self.imageView.frame = CGRectMake(...);
            self.scrollView.contentSize = image.size;
        });
    });
    dispatch_release(downloadQueue); // erst, wenn leer !
}
```

Core Location

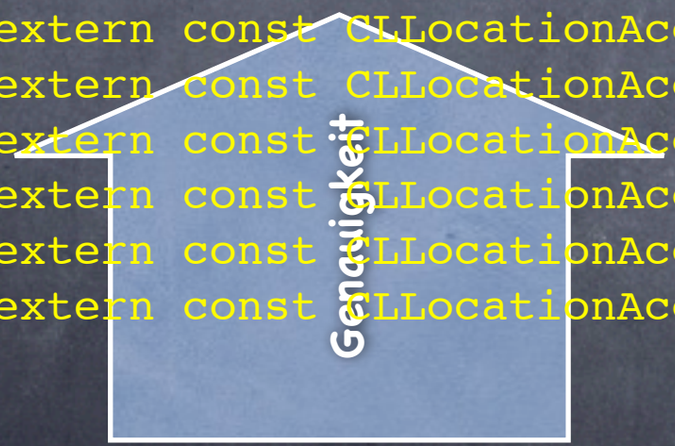
- Positionen auf der Erde spezifizieren
  - non UI
- wichtigste Klasse `CLLocation`

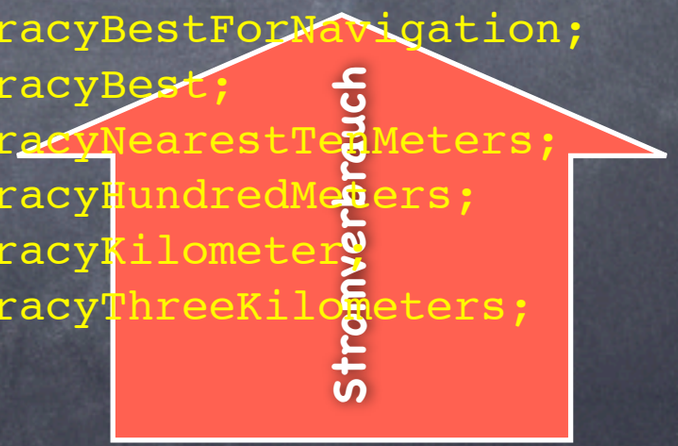
```
@property (readonly) CLLocationCoordinate2D coordinate;
typedef {
    CLLocationDegrees latitude; // double
    CLLocationDegrees longitude;
} CLLocationCoordinate2D;

@property (readonly) CLLocationDistance altitude;
// in Meter, < 0 unter dem Meeresspiegel (sonst oft: ungültig)
```

Core Location

- weitere Properties

```
@property (readonly) CLLocationAccuracy horizontalAccuracy;// in Meter
@property (readonly) CLLocationAccuracy verticalAccuracy; // in Meter
// max. Abweichung von realer Position: <0 ungültige Position
// nicht absolut erfragen, sondern mit diesen Konstanten vergleichen:
extern const CLLocationAccuracy kCLLocationAccuracyBestForNavigation;
extern const CLLocationAccuracy kCLLocationAccuracyBest;
extern const CLLocationAccuracy kCLLocationAccuracyNearestTenMeters;
extern const CLLocationAccuracy kCLLocationAccuracyHundredMeters;
extern const CLLocationAccuracy kCLLocationAccuracyKilometer;
extern const CLLocationAccuracy kCLLocationAccuracyThreeKilometers;
```





Core Location

- Position wird so gut wie möglich ermittelt:
  - Funkmast-Triangulation (nicht sehr genau, aber stromsparend)
  - WiFi Knoten-Lookup (genauer, mehr Stromverbrauch)
  - GPS (am genausten, höchster Stromverbrauch)

Core Location

- weitere Properties

```
@property (readonly) CLLocationSpeed speed;
// Momentanschwindigkeit (m/s) nicht allzu genau, <0 nicht verfügbar

@property (readonly) CLLocationDirection course;
// in Grad, 0 == Norden, im Uhrzeigersinn <0 nicht verfügbar
// nicht auf allen Geräten verfügbar

@property (readonly) NSDate *timestamp;
// von wann ist die Information
```

- Abstand

```
- (CLLocationDistance)distanceFromLocation:(CLLocation *)otherLocation;
```

Core Location

- woher bekommt man eine `CLLocation`?
  - man kann auch direkt Position (`location`)und Richtung (`heading`) erfragen, eher unüblich - stattdessen
  - übliches Vorgehen
    1. prüfen, ob die HW die gewünscht Art der Lokalisierung unterstützt UND ob der Benutzer sie erlaubt hat
    2. einen `CLLocationManager` erzeugen und sich selbst als `delegate` eintragen (`alloc/init`)
    3. den `CLLocationManager` mit der gewünschten Art der Lokalisierungs-Aktualisierung konfigurieren
    4. den Manager starten / stoppen

Core Location

- ad 1.

```
+ (BOOL)locationServicesEnabled;
// hat der Nutzer Positionierung erlaubt?

+ (BOOL)headingAvailable;
// kann das Gerät die Richtung ermitteln
// (hat es einen Kompass)?

+ (BOOL)significantLocationChangeMonitoringAvailable;
// ist es ein Telefon (Funkmast-basiert) ?

+ (BOOL)regionMonitoringAvailable; // nur bestimmte iOS4 devices
+ (BOOL)regionMonitoringEnabled; // vom Nutzer zugelassen ?

@property (copy) NSString *purpose; // wird bei erster
// Verwendung der Lokalisierung angezeigt, sollte man setzen
```

Core Location

- ad 3. mögliche Arten der Lokalisierungs-Aktualisierung

1. Genauigkeitsbasierte Aktualisierung
2. Richtungs (Heading) - Monitoring
3. Aktualisierung bei signifikanten Änderungen der Position
4. Regionen-basierte Aktualisierung

Core Location

1. Genauigkeitsbasierte Aktualisierung: informiere (mich) bei Änderungen der Position

```
@property CLLocationAccuracy desiredAccuracy;
// immer nur so klein wie nötig setzen (s.o.)

@property CLLocationDistance distanceFilter;
// nur wenn Änderung > distanceFilter (Meter)

- (void)startUpdatingLocation; // AN

- (void)stopUpdatingLocation;  // AUS

// immer ausschalten (ggf. auch nur kurzzeitig),
// wenn Ergebnisse nicht gebraucht werden

// delegate-Methode:
- (void)locationManager:(CLLocationManager *)manager
    didUpdateToLocation:(CLLocation *)newLocation
           fromLocation:(CLLocation *)oldLocation;
```

Core Location

2. Richtungs-Monitoring

```
@property CLLocationDegrees headingFilter;
// nur wenn Änderung > headingFilter (Grad)

@property CLHeadingOrientation headingOrientation;
// 0° oben am Gerät, wenn gedreht, z.B. CLDeviceOrientationLandscapeLeft

@property (readonly) CLLocationDirection magneticHeading; // nur Kompass
@property (readonly) CLLocationDirection trueHeading; // inkl. Location
@property (readonly) CLLocationDirection headingAccuracy; // in Grad
@property (readonly) NSDate *timestamp;

- (void)startUpdatingHeading; // AN

- (void)stopUpdatingHeading;  // AUS

// immer ausschalten (ggf. auch nur kurzzeitig),
// wenn Ergebnisse nicht gebraucht werden

// delegate-Methode:
- (void)locationManager:(CLLocationManager *)manager
      didUpdateHeading:(CLHeading *)newHeading; // ohne oldHeading !
```

Core Location

über Fehler bei 1. und 2. wird der delegate informiert:

```
- (void)locationManager:(CLLocationManager *)manager
       didFailWithError:(NSError *)error;
// nicht immer dramatisch, ggf. später beu versuchen
kCLErrorLocationUnknown
// im Moment nicht, aber vielleicht später

kCLErrorDenied
// der Nutzer hat‘s verboten

kCLErrorHeadingFailure
// zu starke magnetisch Interferenzen, nochmal versuchen
```

Core Location

3. Aktualisierung bei signifikanten Änderungen der Position

… wenn ein neuer Funkmast sichtbar wird …

```
- (void)startMonitoringSignificantLocationChanges;  // AN
- (void)stopMonitoringSignificantLocationChanges;   // AUS
// immer ausschalten (ggf. auch nur kurzzeitig),
// wenn Ergebnisse nicht gebraucht werden

// delegate-Methode:
- (void)locationManager:(CLLocationManager *)manager
    didUpdateToLocation:(CLLocation *)newLocation
           fromLocation:(CLLocation *)oldLocation;
```

Funktioniert auch, wenn die App im Hintergrund oder gar nicht läuft ! (?)

Core Location

- Funktioniert auch, wenn die App im Hintergrund oder gar nicht läuft ! (?)
  - wie das?
  - sie wird (bei signif. Änderungen) in den Vordergrund geholt oder sogar neu gestartet und `application:didFinishLaunchingWithOptions:`
  - Optionen sind ein Dictionary, in diesem Fall enthält es den Key: `UIApplicationLaunchOptionsLocationKey`
  - einen `CLLocationManager` erzeugen und nach der `location` befragen

Core Location

4. Regionen-basierte Aktualisierung

   … wenn ich ein Gebiet betrete/verlasse …

```
- (void)startMonitoringForRegion:(CLRegion *)
                  desiredAccuracy:(CLLocationAccuracy);  // AN

- (void)stopMonitoringForRegion:(CLRegion *);           // AUS
```

```
// delegate-Methoden:
- (void)locationManager:(CLLocationManager *)manager
         didEnterRegion:(CLRegion *)region;

- (void)locationManager:(CLLocationManager *)manager
          didExitRegion:(CLRegion *)region;

- (void)locationManager:(CLLocationManager *)manager
      monitoringDidFailForRegion:(CLRegion *)region
                       withError:(NSError *)error;
```

Funktioniert auch, wenn die App im Hintergrund oder gar nicht läuft.

Core Location

- die registrierten Regionen sind persistent:

```
@property (readonly) NSSet *monitoredRegions;
// CLLocationManager property
```

- Regionen erzeugen

```
- (id)initCircularRegionWithCenter:(CLLocationCoordinate2D)center
                            radius:(CLLocationDistance)radius
                        identifier:(NSString *)identifier
// aus gleich benannten Properties auslesbar
```

- benutzt NICHT GPS (sondern wohl Funkmasten)
- es gibt einen maximalen Radius

```
@property (readonly) CLLocationDistance
maximumRegionMonitoringDistance;
// < 0: Region-Monitoring nicht verfügbar
```

Map Kit

- Google Maps anzeigen

  http://code.google.com/intl/de-DE/apis/maps/iphone/terms.html

- mit Annotationen (`MKAnnotation`)
  - dargestellt als Pins (`MKPinAnnotationView`) o.ä. (`MKAnnotationView`)
  - können sog. `Callout` haben (bei Klick)
- mit sog. Overlays (`MKOverlay`)

Map Kit

- `MKMapView` (im IB am Stück in der Palette)
  - zeigt Karte mit Annotationen (und Overlays s.u.)

    ```
    @property (nonatomic, readonly) NSArray* annotations;
    ```

  - enthaltene Objekte müssen Protokoll `MKAnnotation` implementieren:

    ```
    @protocol MKAnnotation <NSObject>
    @property (readonly) CLLocationCoordinate2D coordinate;
    @optional
    @property (readonly) NSString *title;
    @property (readonly) NSString *subtitle;
    @end
    typedef {
        CLLocationDegrees latitude;   // double
        CLLocationDegrees longitude;  // double
    } CLLocationCoordinate2D;
    ```

Map Kit

- `annotations` ist immutable:

```
- (void)addAnnotation:(id <MKAnnotation>)annotation;
- (void)addAnnotations:(NSArray *)annotations;
- (void)removeAnnotation:(id <MKAnnotation>)annotation;
- (void)removeAnnotations:(NSArray *)annotations;
```

- `MKMapView` hat einen

```
@property(nonatomic, assign) id<MKMapViewDelegate> delegate;
```

- dieser sollte die Anzahl der Annotations auf die (aktuell) sichtbaren

```
/*MKMapView*/ @property (readonly) MKMapRect visibleRect;
```

beschränken (Performance/Speicher)

Map Kit

- in den MKMapViewDelegate Methoden

```
- (void)mapView:(MKMapView *)sender
   regionDidChangeAnimated:(BOOL)animated;

// auch “will”-Version, aber Vorsicht: wird beim Scrollen
// ständig gerufen

- (void)mapView:(MKMapView *)sender
   regionWillChangeAnimated:(BOOL)animated;
```

- z.B. mit

```
MKMapPoint annotationPoint =
    MKMapPointForCoordinate(annotation.coordinate);
if (MKMapRectContainsPoint(mapView.visibleRect, annotationPoint))
{ ... }
```

Map Kit

- Annotationen werden per default als Pins (`MKPinAnnotationView`) dargestellt
- eigene Ableitungen von `MKAnnotationView` möglich
- bereits die Basisklasse `MKAnnotationView` hat Properties

```
@property (retain) id <MKAnnotation> annotation;
// die Annotation

@property (retain) UIImage *image; // dieses Bild anstelle des Pins

@property CGPoint centerOffset; // durch image markierter Punkt

@property (retain) UIView *leftCalloutAccessoryView;
// z.B. ein UIImageView

@property (retain) UIView *rightCalloutAccessoryView;
// z.B. eine Disclosure Button

@property BOOL canShowCallout;
@property BOOL enabled;
// NO: ignoriert Klick, kein callout, kein Aufruf von der
// MKMapViewDelegate-Methode mapView:didSelectAnnotationView:

@property BOOL draggable;
// nur wenn die Annotation setCoordinate: implementiert
```

Map Kit

- bei Klick auf eine Annotation
  - Aufruf der MKMapViewDelegate-Methode

    ```
    - (void)mapView:(MKMapView *)sender
        didSelectAnnotationView:(MKAnnotationView *)aView;
    ```

    darin z.B. neuen ViewController auf den NavivationController packen oder Callout vorbereiten `left/rightCalloutAccessoryView`
  - automatisch: falls beim zugeordneten `MKAnnotationView canShowCallout==YES`: fertigen Callout anzeigen

Map Kit

- Erzeugung und Zuordnung von `MKAnnotationView`'s
  - in der `MKMapViewDelegate`-Methode

```
- (MKAnnotationView *)mapView:(MKMapView *)sender
            viewForAnnotation:(id <MKAnnotation>)annotation
```

  - falls nicht implementiert: returns `MKPinAnnotationView`

Map Kit

- ähnlich wie Erzeugung von UITableViewCells:

```
- (MKAnnotationView *)mapView:(MKMapView *)sender
            viewForAnnotation:(id <MKAnnotation>)annotation {
  MKAnnotationView *aView = [sender
            dequeueReusableAnnotationViewWithIdentifier: IDENT];
  if (!aView) {
   aView = [[[MKPinAnnotationView alloc] // oder Ableitung
                 initWithAnnotation:annotation
                 reuseIdentifier:IDENT] autorelease];
        // setze canShowCallout auf YES und
        // (eher unüblich) setze aView's callout
        // accessories bereits hier
  }
  aView.annotation = annotation; // IMMER auch wenn grad neu verknüpft
  // setze accessory views und/oder title/subtitle
  // oder setze sie auf nil mapView:didSelectAnnotationView:
  // sollte sie dann später setzen
  return aView;
}
```

Map Kit

- Setzen der AccessoryViews erst bei Auswahl z.B.

```
- (void)mapView:(MKMapView *)sender
   didSelectAnnotationView:(MKAnnotationView *)aView {
    if ([aView.leftCalloutAccessoryView isKindOfClass:[UIImageView class]]) {
        UIImageView *imageView = (UIImageView *)aView.leftCalloutAccessoryView;
        dispatch_queue_t downloader =
                    dispatch_queue_create(“callout downloader”, NULL);
        dispatch_async(downloader,
            ^{
                UIImage *theImage = ...; // download theImage
                dispatch_async(dispatch_get_main_queue(),
                               ^{
                                   imageView.image = theImage;
                               });
            });
        dispatch_release(downloader);
    }
```