

Praktikum Entwicklung von Mediensystemen mit iOS

SS 2012

Prof. Dr. Michael Rohs
michael.rohs@ifi.lmu.de
MHCI Lab, LMU München

Today

- Alerts, Action Sheets, text input
- Table views
- Touch input
- Data management
- Location and sensors
- Networking
- Exercise 2

Timeline

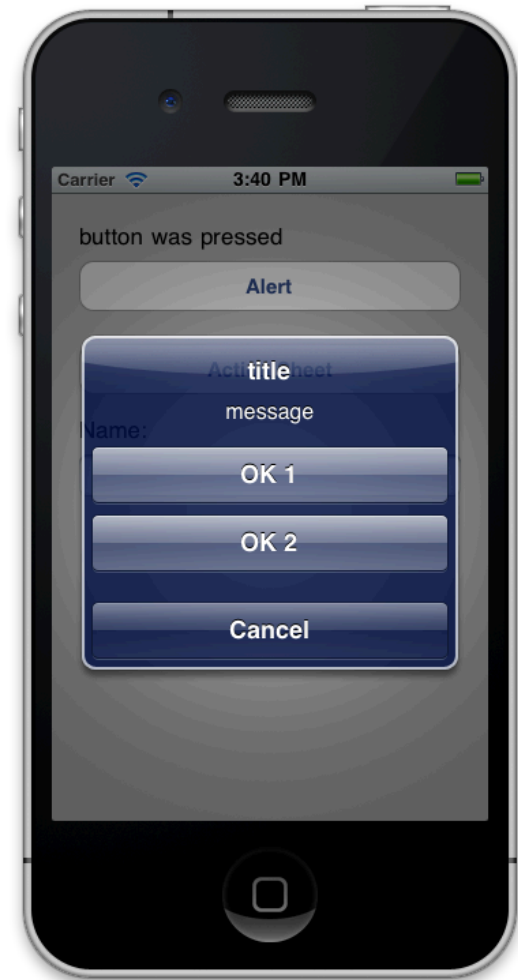
#	Date	Topic
	19.4.	Introduction & Brainstorming future mobile concepts
1	3.5.	Video watching, Introduction to iOS
	10.5.	no class (CHI Konferenz)
	17.5.	no class (Christi Himmelfahrt)
2	24.5.	More on iOS
3	31.5.	Concept finalization, paper prototyping
	7.6.	no class (Frohnleichnam)
	14.6.	Paper prototyping test, start of software prototype
5	21.6.	
6	28.6.	Think aloud study of software prototype
7	5.7.	
8	12.7.	Completion of software prototype
9	19.7.	Final presentation

MORE ON IOS

Showing a UIAlertView

```
-(IBAction) alertButtonPressed:(id)sender;
{
    UIAlertView *alert = [[UIAlertView alloc]
        initWithTitle:@"title"
        message:@"message"
        delegate:nil
        cancelButtonTitle:@"Cancel"
        otherButtonTitles:@"OK 1", @"OK 2", nil];

    [alert show];
    [alert release];
}
```

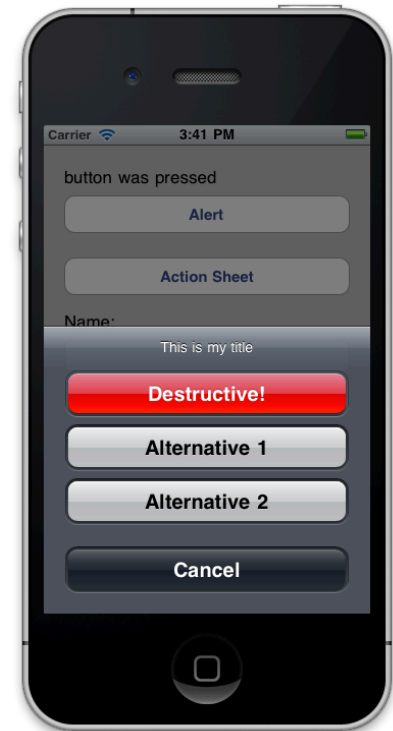


Showing an Action Sheet

- Action sheet steps
 - Implement `UIActionSheetDelegate` protocol
 - Construct, `showInView`, release
 - Implement delegate method `clickedButtonAtIndex`

- Code

```
- (IBAction)actionSheetButtonPressed:(id)sender {
    UIActionSheet *actionSheet = [[UIActionSheet alloc]
        initWithTitle:@"This is my title" delegate:self
        cancelButtonTitle:@"Cancel"
        destructiveButtonTitle:@"Destructive!"
        otherButtonTitles:@"Alternative 1", @"Alternative 2", nil];
    [actionSheet showInView:self.view];
    [actionSheet release];
}
```

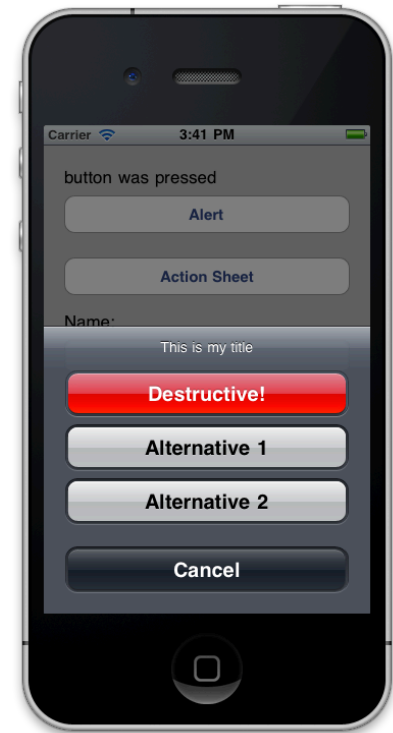


Process Action Sheet Events

- Action sheet steps
 - Declare `UIActionSheetDelegate` protocol
 - Construct, `showInView`, release
 - Implement delegate method `clickedButtonAtIndex`

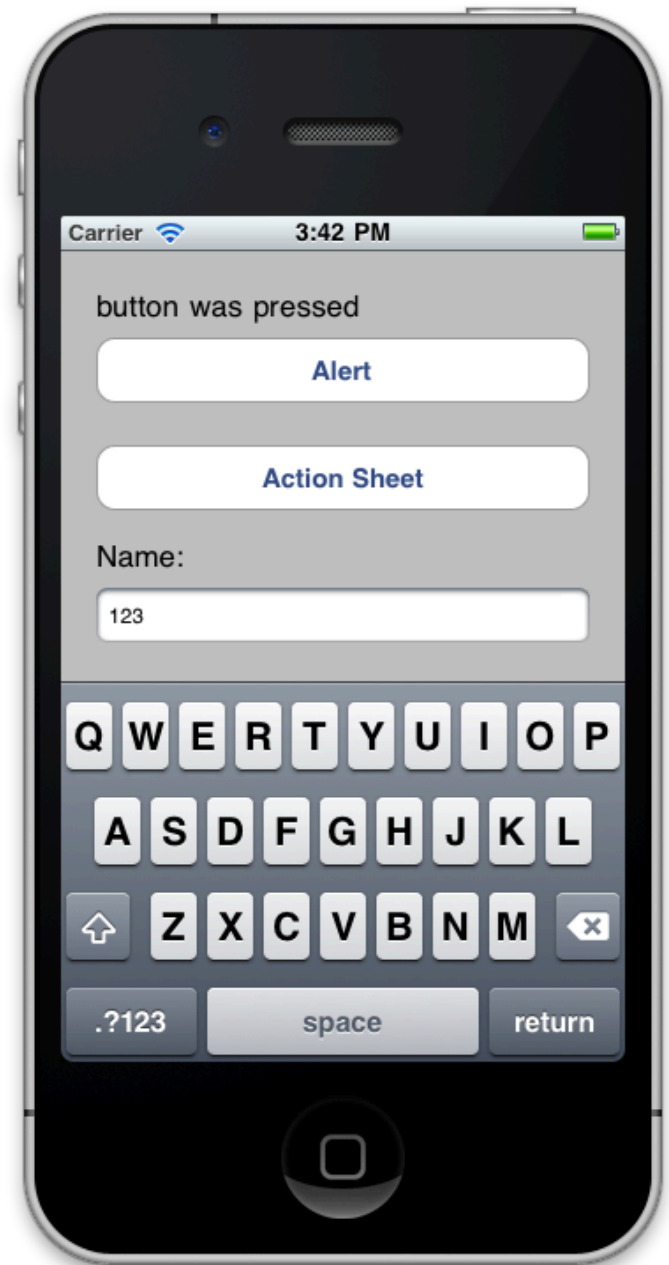
- Code

```
- (void)actionSheet:(UIActionSheet *)actionSheet
    clickedButtonAtIndex:(NSInteger)buttonIndex
{
    NSLog(@"actionSheet: clickedButtonAtIndex %d", buttonIndex);
}
```



Text Input

- Add UITextField in Storyboard
- Add member variable and property to .h, synthesize in .m
- Declare UITextFieldDelegate in .h
- Implement delegate methods in .m, set label text on end editing
- Set delegate in ViewController viewDidLoad method or set delegate in Storyboard

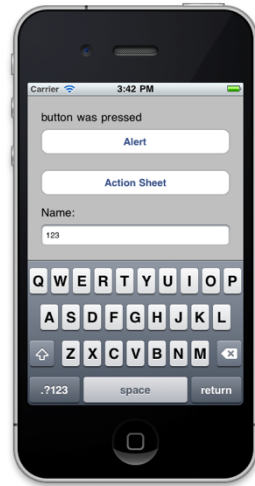


Process Text Input (1/2)

- ```
- (BOOL)textFieldShouldBeginEditing:(UITextField *)textField {
 NSLog(@"textFieldShouldBeginEditing: %@", textField.text);
 return YES;
}

- (void)textFieldDidBeginEditing:(UITextField *)textField {
 NSLog(@"textFieldDidBeginEditing: %@", textField.text);
}

- (BOOL) textFieldShouldReturn:(UITextField*)textField {
 NSLog(@"textFieldShouldReturn: %@", textField.text);
 [textField resignFirstResponder];
 return YES;
}
```



# Process Text Input (2/2)

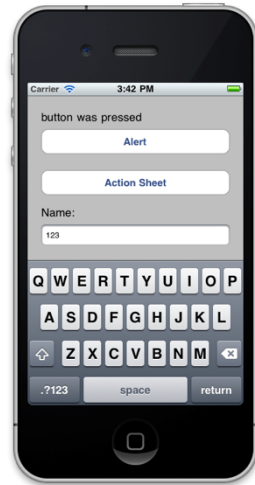
```
- (BOOL)textFieldShouldEndEditing:(UITextField *)textField {
 NSLog(@"textFieldShouldEndEditing: %@", textField.text);
```

```
 UILabel *label = (UILabel*)[self.view viewWithTag:100];
 label.text = textField.text;
 return YES;
```

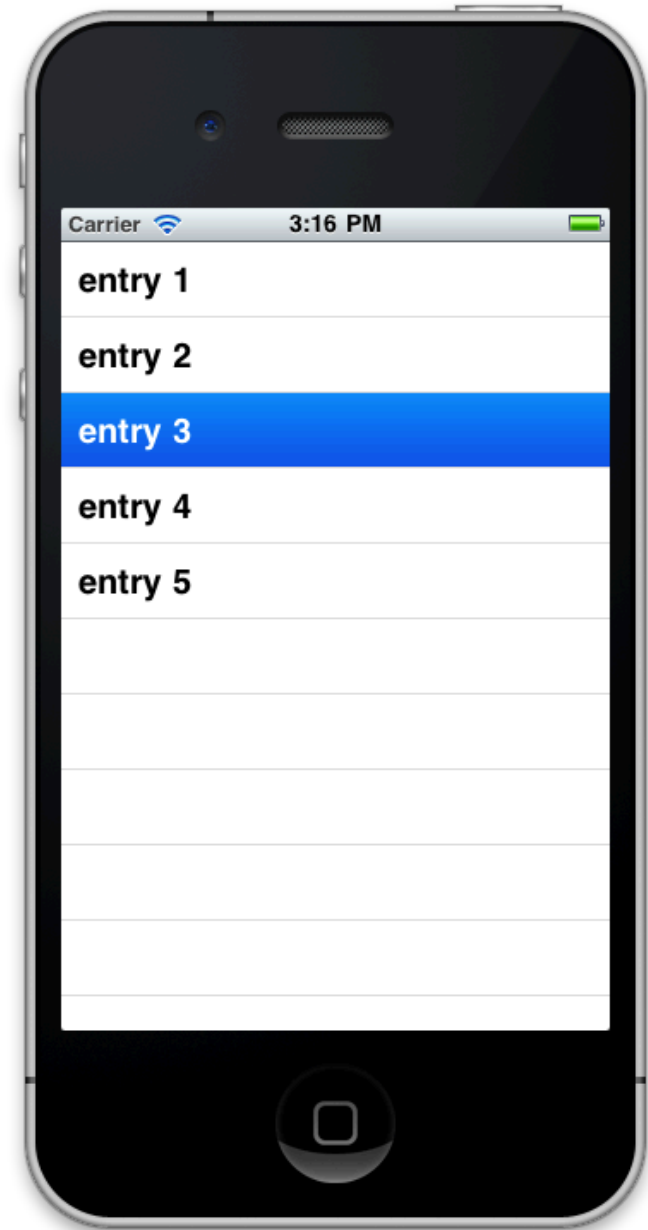
```
}
```

```
- (void)textFieldDidEndEditing:(UITextField *)textField {
 NSLog(@"textFieldDidEndEditing: %@", textField.text);
```

```
}
```



# UITABLEVIEW CONTROLLER



# UIViewController subclasses

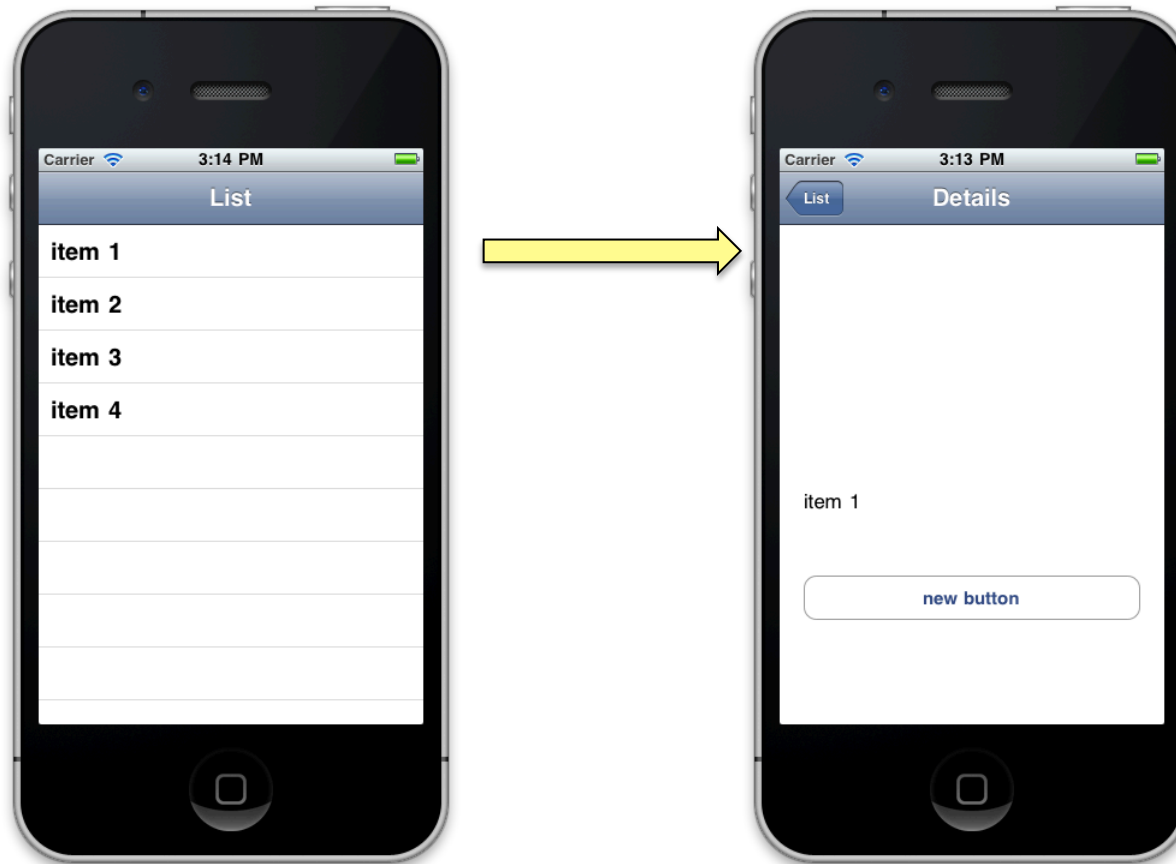
- View lifecycle
  - (void)viewDidLoad
  - (void)viewDidUnload
- View events
  - (void) viewWillAppear:(BOOL)animated
  - (void) viewWillDisappear:(BOOL)animated
  - (void) viewDidAppear:(BOOL)animated
  - (void) viewDidDisappear:(BOOL)animated
- Rotation settings and events
  - interfaceOrientation property
    - shouldAutorotateToInterfaceOrientation:
- many more... → see documentation

# UITableViewDataSource (Protocol)

- Configuring a Table View
  - tableView:cellForRowAtIndexPath: required method
  - numberOfSectionsInTableView:
  - tableView:numberOfRowsInSection: required method
  - sectionIndexTitlesForTableView:
  - tableView:sectionForSectionIndexTitle:atIndex:
  - tableView:titleForHeaderInSection:
  - tableView:titleForFooterInSection:
- Inserting or Deleting Table Rows
  - tableView:commitEditingStyle:forRowAtIndexPath:
  - tableView:canEditRowAtIndexPath:
- Reordering Table Rows
  - tableView:canMoveRowAtIndexPath:
  - tableView:moveRowAtIndexPath:toIndexPath:

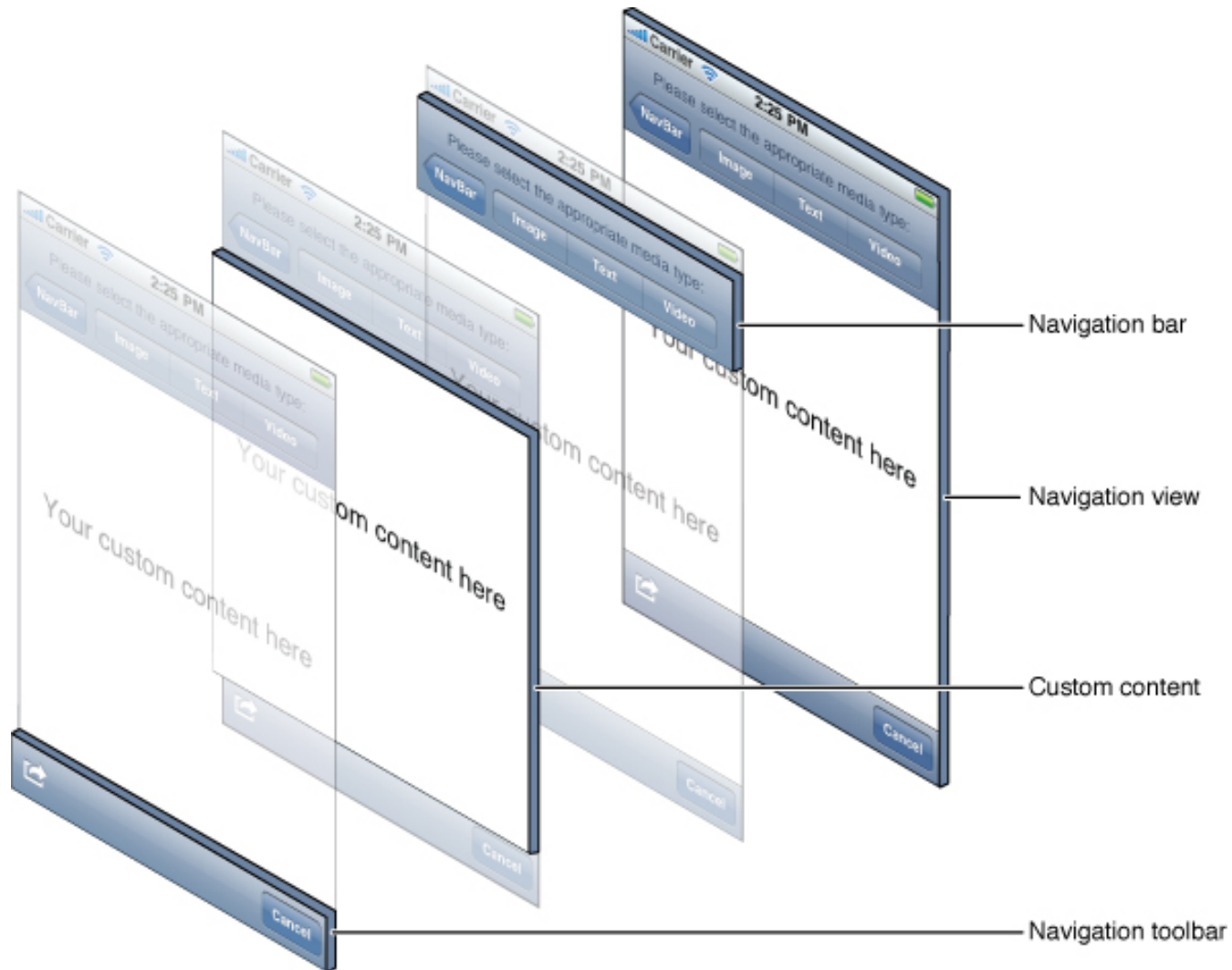
# UITableViewDelegate (Protocol)

- Configuring Rows for the Table View
  - `tableView:heightForRowAtIndexPath:`
- Managing Accessory Views
  - `tableView:accessoryButtonTappedForRowWithIndexPath:`
- Managing Selections
  - `tableView:{will,did}SelectRowAtIndexPath:`
  - `tableView:{will,did}DeselectRowAtIndexPath:`
- Modifying the Header and Footer of Sections
  - `tableView:viewFor{Header,Footer}InSection:`
  - `tableView:heightFor{Header,Footer}InSection:`
- Editing Table Rows
- Reordering Table Rows



# VIEW STACK

# Navigation Controller Views



Source: [http://developer.apple.com/library/ios/#documentation/uikit/reference/UINavigationController\\_Class/Reference/Reference.html](http://developer.apple.com/library/ios/#documentation/uikit/reference/UINavigationController_Class/Reference/Reference.html)



# TOUCH INPUT

# Touch Input

- Overwrite methods in UIView or UIImageView:
  - (void)touchesBegan:(NSSet \*)touches withEvent:(UIEvent \*)event  
{  
    UITouch \*touch = [touches anyObject];  
    CGPoint p = [touch locationInView:self];  
    traceCount = 0;  
    trace[traceCount++] = p;  
    [self updateDisplay];  
}
  - (void)touchesMoved:(NSSet \*)touches withEvent:(UIEvent \*)event;
  - (void)touchesEnded:(NSSet \*)touches withEvent:(UIEvent \*)event;
  - (void)touchesCancelled:(NSSet \*)touches withEvent:(UIEvent \*)event;

# Accessing Application Directories

- Application sandbox: can only access own app folder

```
NSString *homeDir = NSHomeDirectory();
```

```
NSString *tmpDir = NSTemporaryDirectory();
```

```
NSArray *paths =
```

```
 NSSearchPathForDirectoriesInDomains(NSDocumentDirectory,
 NSUserDomainMask, YES);
```

```
 path = [paths objectAtIndex:0];
```

- Accessing data bundled as an application resource

```
NSString *fileName = [homeDir
```

```
 stringByAppendingPathComponent:@"Test.app/mydata.dat"];
```

# DATA MANAGEMENT

# Accessing Application Directories

- Application sandbox: can only access own app folder

```
NSString *homeDir = NSHomeDirectory();
```

```
NSString *tmpDir = NSTemporaryDirectory();
```

```
NSArray *paths =
```

```
 NSSearchPathForDirectoriesInDomains(NSDocumentDirectory,
 NSUserDomainMask, YES);
```

```
 path = [paths objectAtIndex:0];
```

- Accessing data bundled as an application resource

```
NSString *fileName = [homeDir
```

```
 stringByAppendingPathComponent:@"Test.app/mydata.dat"];
```

# Loading and Saving Binary Data

- NSData is a container for bytes

- Loading arbitrary binary data

```
NSData *d = [[NSData alloc] initWithContentsOfFile:fileName];
```

```
NSMutableData *m = [NSData dataWithContentsOfFile:fileName];
```

- Accessing the data

```
const char* b = [d bytes]; // d is immutable → cannot be modified
```

```
char* c = [m mutableBytes]; // m is mutable → can be modified
```

- Saving arbitrary binary data

```
[d writeToFile:fileName atomically:YES];
```

- Appending to mutable data object

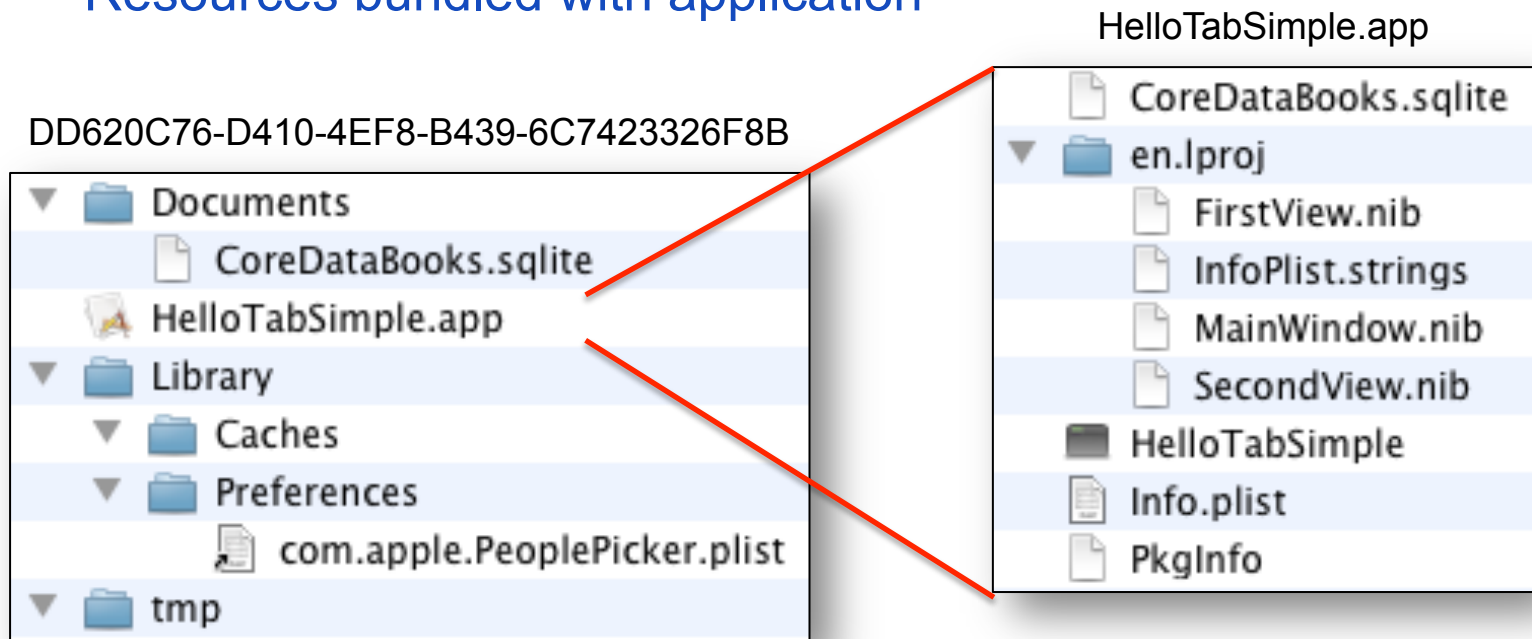
```
[m appendBytes:myBytes length:myBytesCount];
```

# Binary Loading & Saving Code Snippet

```
NSString *fileName = @"./.../myTestFile.dat";
NSData *d = [[NSData alloc] initWithContentsOfFile:fileName];
const char* b = [d bytes];
// use the data, cannot modify
[d release];
NSMutableData *m = [[NSMutableData alloc]
 initWithContentsOfFile:fileName];
char* c = [m mutableBytes];
c[0] = 42; // modify the data (direct access to data)
char *myBytes = "123";
int myBytesCount = strlen(myBytes);
[m appendBytes:myBytes length:myBytesCount];
[m writeToFile:fileName atomically:YES];
[m release];
```

# Directory Structure of an App

- Each app has a unique identifier
  - Identifier is name of directory in “Applications” directory
- .app is a directory itself
  - Cannot be modified (signed)
  - Resources bundled with application





# Copying Resources to Documents Directory

- Copying a resource file to documents directory

```
NSArray *paths = NSSearchPathForDirectoriesInDomains
 (NSDocumentDirectory, NSUserDomainMask, YES);
NSString *documentPath = ([paths count] > 0) ? [paths objectAtIndex:0] : nil;

NSString *filePath = [documentPath stringByAppendingPathComponent:
 @"CoreDataBooks.sqlite"];

NSFileManager *fileManager = [NSFileManager defaultManager];
if (![fileManager fileExistsAtPath:filePath]) {
 NSString *defaultFilePath = [[NSBundle mainBundle]
 pathForResource:@"CoreDataBooks" ofType:@"sqlite"];
 if (defaultFilePath) {
 [fileManager copyItemAtPath:defaultFilePath toPath:filePath error:NULL];
 }
}
```

# Loading XML Data

- XML data and property lists for structured data
- Predefined elements dict, array, string, key, integer, etc.
- Example (a dictionary containing an array of dictionaries)

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
 <dict>
 <key>images</key>
 <array>
 <dict>
 <key>title</key><string>My Image Title</string>
 <key>image</key><string>MyImage.png</string>
 </dict>
 <dict>
 <key>title</key><string>Another Title</string>
 <key>image</key><string>AnotherImage.png</string>
 </dict>
 </array>
 </dict>
</plist>
```

# Loading and Saving Object Hierarchies

- Declaring objects as archiveable by implementing NSCopying protocol
  - initWithCoder, encodeWithCoder
- Handle archiving in these methods
  - All objects handled by coder need to conform to NSCopying
- NSKeyedArchiver to save object hierarchy
- NSKeyedUnarchiver to load object hierarchy

# Declaring Classes as Archiveable

```
@interface MyClass : NSObject <NSCopying> {
 NSString *lastName;
 NSMutableArray *firstNames;
}
- (id) initWithCoder:(NSCoder *)decoder {
 self = [super init];
 self.lastName = [decoder decodeObjectForKey:@"lastName"];
 self.firstNames = [decoder decodeObjectForKey:@"firstNames"];
 return self;
}
- (void) encodeWithCoder:(NSCoder *)encoder {
 [encoder encodeObject:lastName forKey:@"lastName"];
 [encoder encodeObject:firstNames forKey:@"firstNames"];
}
```

# Saving Object Hierarchies

- Archiving (simple version, one root)

```
[NSKeyedArchiver archiveRootObject:myRoot toFile:myFile];
```

- Archiving (complex version, multiple roots)

```
NSMutableData *data = [[NSMutableData alloc] init];
```

```
NSKeyedArchiver *archiver = [[NSKeyedArchiver alloc]
 initWithWritingWithMutableData:data];
```

```
[archiver encodeObject:myRoot1 forKey:@"myRoot1"];
```

```
[archiver encodeObject:myRoot2 forKey:@"myRoot2"];
```

```
[archiver finishEncoding];
```

```
[data writeToFile:myFile atomically:YES];
```

```
[archiver release];
```

```
[data release];
```

# Loading an Object Hierarchy

- Unarchiving an object hierarchy

```
self.object = [NSKeyedUnarchiver unarchiveObjectWithFile:fileName];
```

# CoreData

- Creation of data models (entities, relationships)
- Persistent storage of object graphs
- Undo management
- Version management
  
- Good for complex data models
- Data storage can be database or flat file

# Core Data Stack

- Persistent storage (e.g. database)
- Object store maps records to objects
- Store coordinator aggregates stores
- Managed object model contains entity descriptions
- Managed object context contains managed objects
- Managed objects are part of the persistent object graph

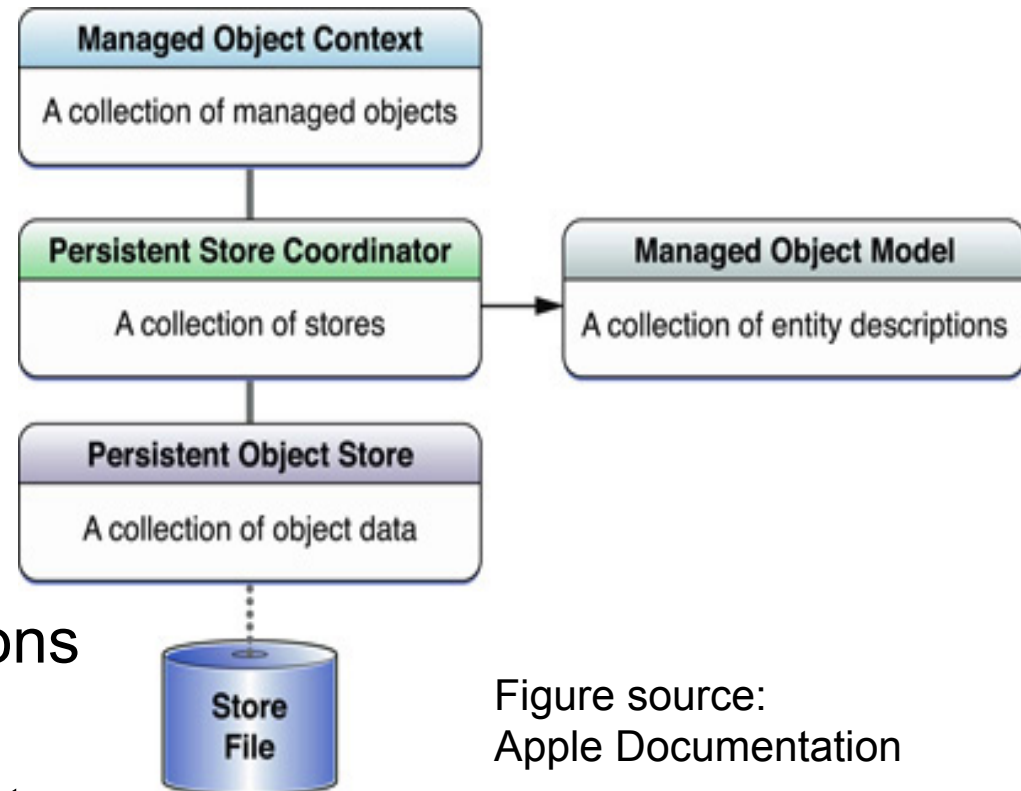


Figure source:  
Apple Documentation



# Managed Object

- Objective C object that represents a data record of the persistent store (e.g. row of a database)
- Has a reference to its entity description
- Part of a managed object context

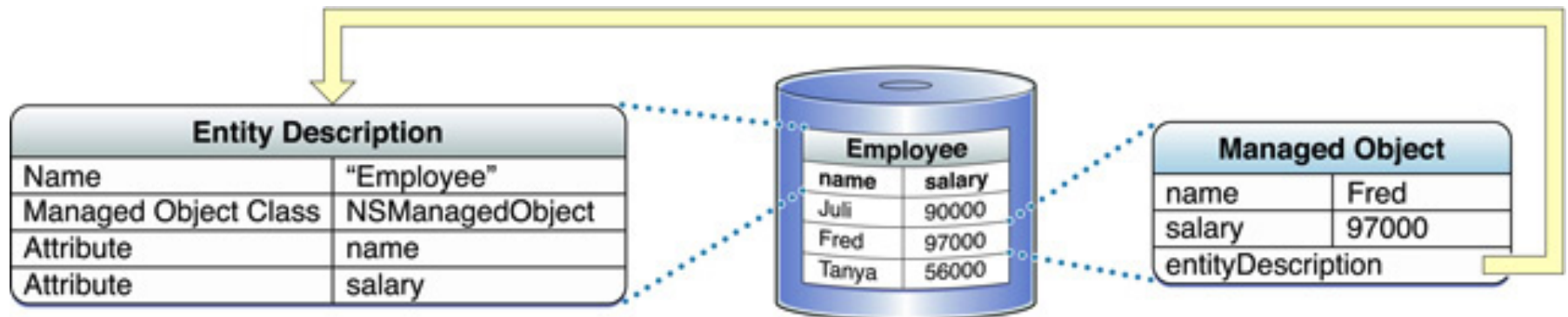
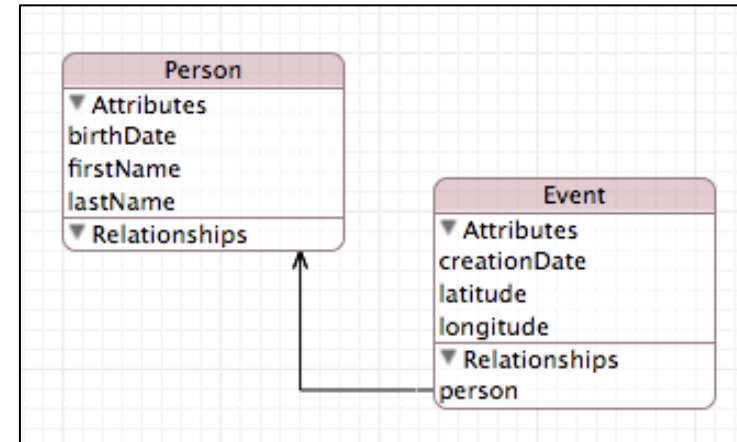
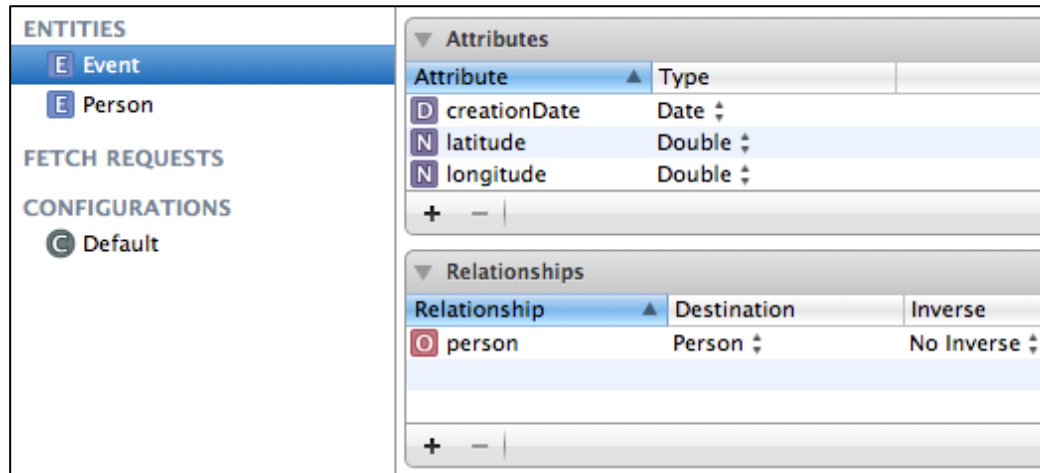


Figure source: Apple Documentation

# Defining the Object Model

- Xcode data model



- Automatic creation of model class

```
@interface Event : NSObject {
@property (nonatomic, strong) NSNumber * latitude;
@property (nonatomic, strong) NSNumber * longitude;
@property (nonatomic, strong) NSDate * creationDate;
@property (nonatomic, strong) Person * person;
@end
```

# Creating NSPersistentStoreCoordinator

```
NSURL *storeUrl = [NSURL fileURLWithPath:
 [[self applicationDocumentsDirectory]
 stringByAppendingPathComponent:@"Locations.sqlite"]];
```

```
managedObjectModel = [NSManagedObjectModel
 mergedModelFromBundles:nil];
```

```
persistentStoreCoordinator = [[NSPersistentStoreCoordinator alloc]
 initWithManagedObjectModel:[self managedObjectModel]];
```

```
[persistentStoreCoordinator
 addPersistentStoreWithType:NSSQLiteStoreType
 configuration:nil
 URL:storeUrl
 options:nil
 error:nil];
```

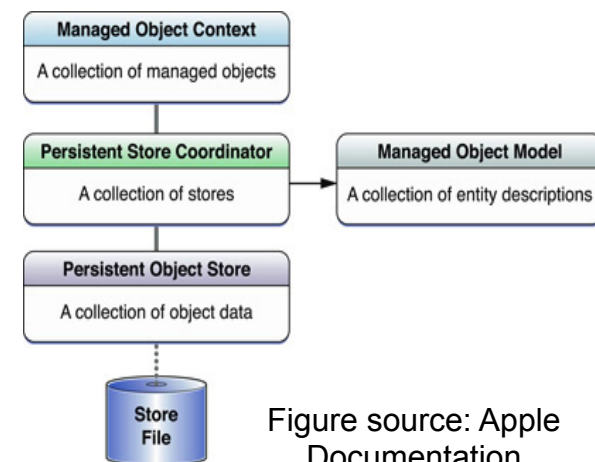


Figure source: Apple Documentation

# NSManagedObjectContext

- Creation

```
managedObjectContext = [[NSManagedObjectContext alloc] init];
[managedObjectContext setPersistentStoreCoordinator: coordinator];
```

- Persistent saving of object graph

```
NSError *error;
if (![self managedObjectContext
 save:&error]) {
 // handle error
}
```

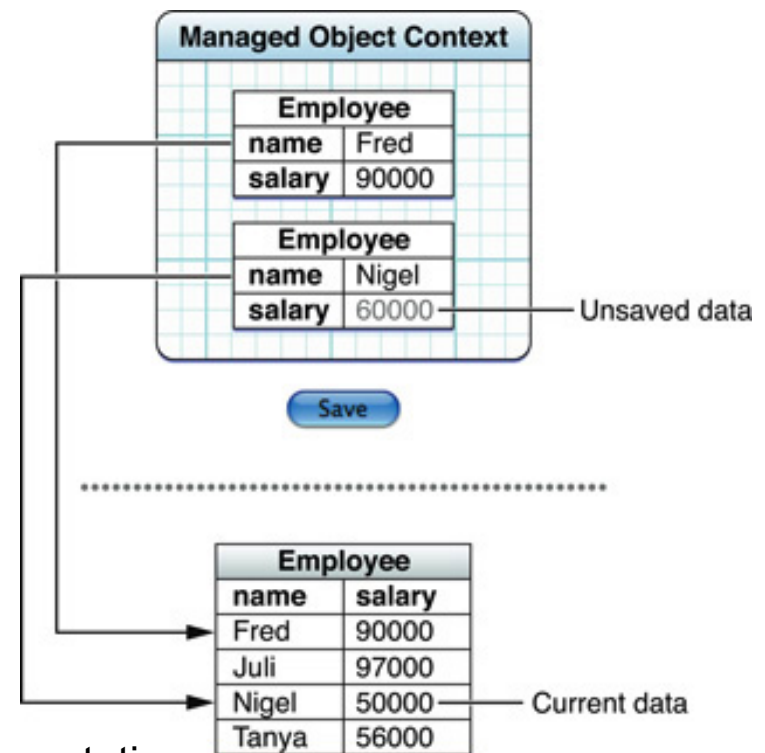


Figure source: Apple Documentation

# Loading objects from store

- Fetch requests

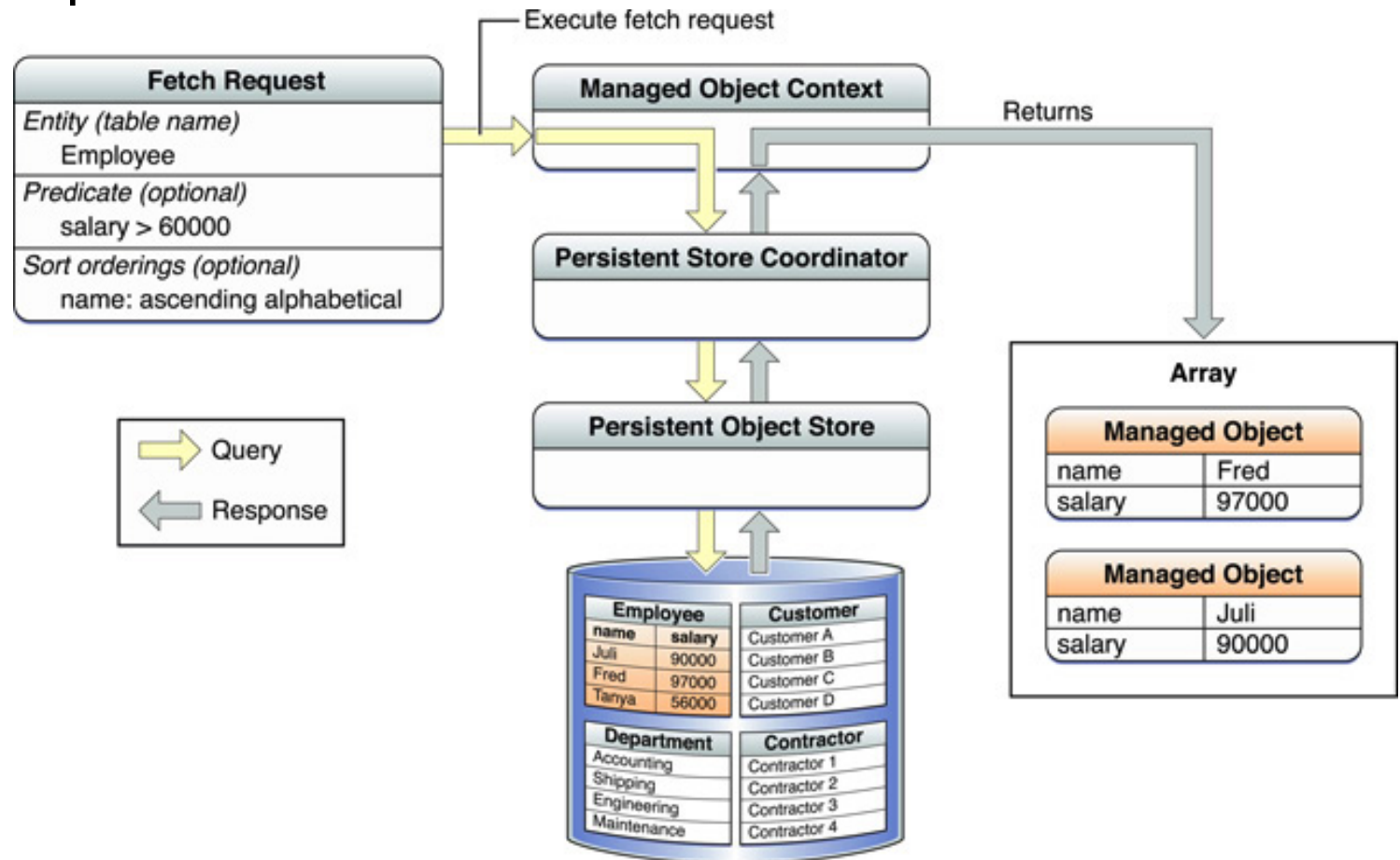


Figure source: Apple Documentation

# Loading objects from store

```
NSFetchRequest *request = [[NSFetchRequest alloc] init];
NSEntityDescription *entity = [NSEntityDescription entityForName:@"Event"
 inManagedObjectContext:managedObjectContext];
[request setEntity:entity];

NSSortDescriptor *sortDescriptor = [[NSSortDescriptor alloc]
initWithKey:@"creationDate" ascending:NO];
NSArray *sortDescriptors = [NSArray arrayWithObject:sortDescriptor];
[request setSortDescriptors:sortDescriptors];
[sortDescriptor release];

NSError *error = nil;
NSArray *results = [managedObjectContext
 executeFetchRequest:request error:&error];
NSMutableArray *mutableResults = [results mutableCopy];
```

# Removing an object from the Context

- “release” does not remove managed object
- need to explicitly remove from context and commit  
[managedObjectContext deleteObject:eventToDelete];  
[managedObjectContext save:nil];
- also remove from local data structure and GUI element  
[eventsArray removeObjectAtIndex:indexPath.row];  
[tableView deleteRowsAtIndexPaths:[NSArray  
    arrayWithObject:indexPath]  
    withRowAnimation:YES];

# Creating a new managed object

```
Event *event = (Event *)[NSEntityDescription
 insertNewObjectForEntityForName:@"Event"
 inManagedObjectContext:managedObjectContext];
```

```
[event setLatitude:<my latitude>];
[event setLongitude:<my longitude>];
[event setCreationDate:[NSDate date]];
```

```
// Save persistently
```

```
[managedObjectContext save:nil];
```

```
// Insert into array
```

```
[eventsArray insertObject:event atIndex:0];
```



# LOCATION & SENSORS

# Location

- Location manager provides location information

```
CLLocationManager *locationManager;
```

- Configuration

```
locationManager = [[CLLocationManager alloc] init];
```

```
locationManager.delegate = self;
```

```
locationManager.desiredAccuracy = [[setupInfo
objectForKey:@"SetupInfoKeyAccuracy"] doubleValue];
```

```
[locationManager startUpdatingLocation];
```

- Update location

```
(void)locationManager:(CLLocationManager *)manager
didUpdateToLocation:(CLLocation *)newLocation fromLocation:
(CLLocation *)oldLocation { ... }
```

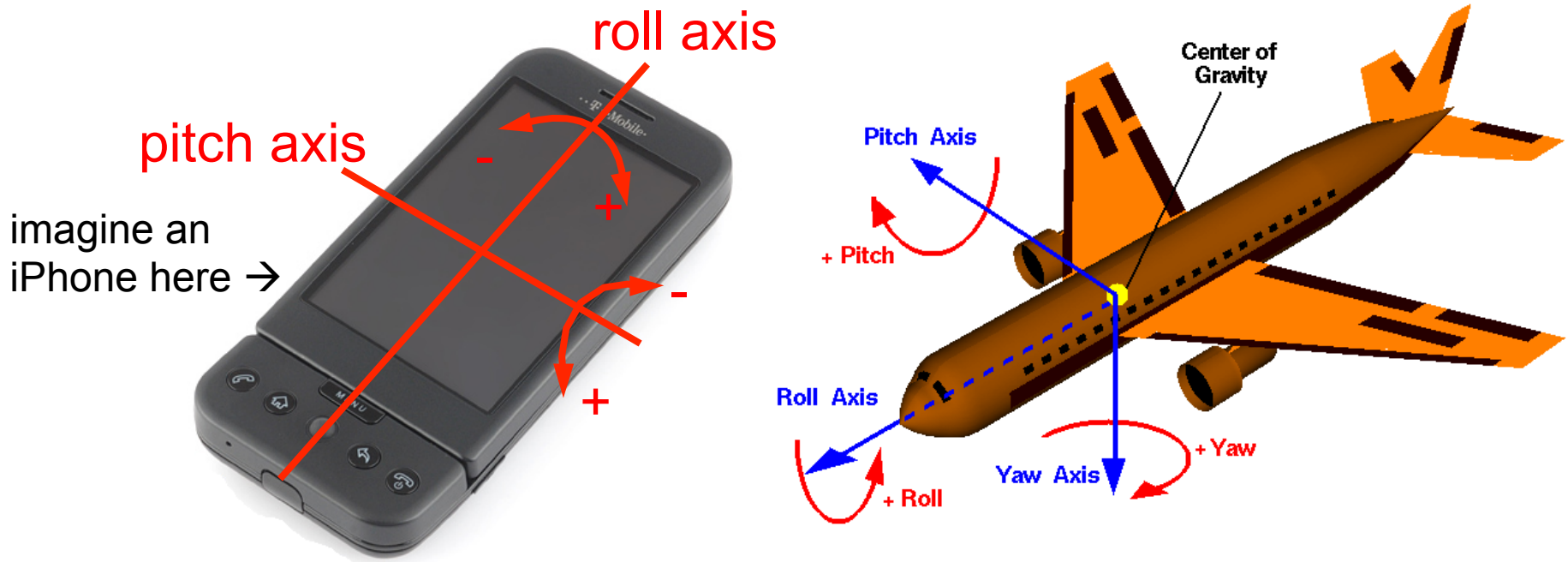
- Details: LocateMe example application

# Latitude and Longitude

- Location manager state  
    `manager.locationServicesEnabled`
- Location manager desiredAccuracy constants
  - `kCLLocationAccuracyBestForNavigation`
  - `kCLLocationAccuracyBest`
  - `kCLLocationAccuracyNearestTenMeters`
  - `kCLLocationAccuracyHundredMeters`
  - `kCLLocationAccuracyKilometer`
  - `kCLLocationAccuracyThreeKilometers`
- `CLLocation`
- `CLLocationDegrees`
- `CLLocationDistance` (from location ... to location)

# Sensor Coordinate Systems

“Orientation Sensor” = Accelerometer + Magnetometer



Axis	Zero position	Range	Details
Yaw	north	$-\pi.. \pi$	0=north, $-\pi/2$ =east, $\pm\pi$ =south, $\pi/2$ =west
Pitch	horizontal	$-\pi/2.. \pi/2$	0=horizontal, >0: up, <0: down
Roll	horizontal	$-\pi.. \pi$	0=horizontal, >0: right, <0: left

# Accelerometer (since iOS 2.0)

- UIAccelerometer

```
UIAccelerometer *acc;
```

- Configure accelerometer

```
acc = [UIAccelerometer sharedAccelerometer];
```

```
acc.updateInterval = 1.0; // sec
```

```
acc.delegate = self;
```

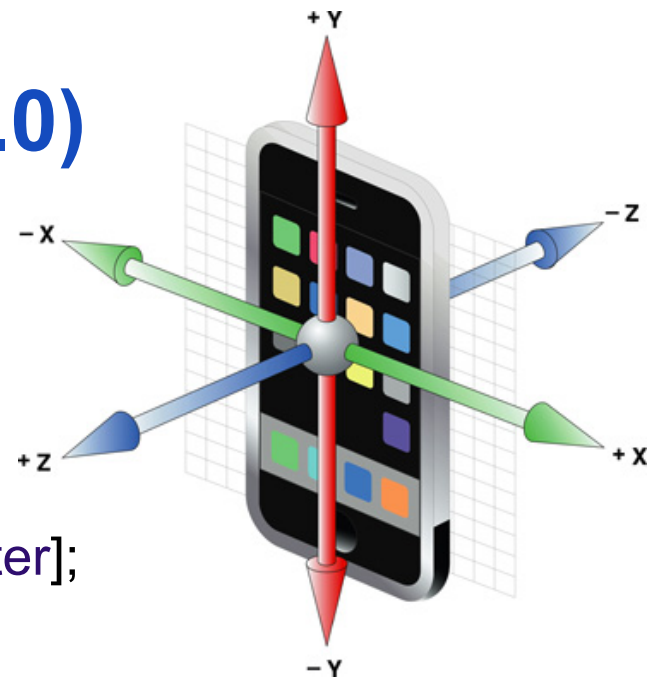
- Receive updates

```
-(void)accelerometer:(UIAccelerometer *)a
 didAccelerate:(UIAcceleration *)acceleration
```

```
{
```

```
 NSLog(@"%.3f, %.3f, %.3f, %f",
 acceleration.x, acceleration.y,
 acceleration.z, acceleration.timestamp);
```

```
}
```



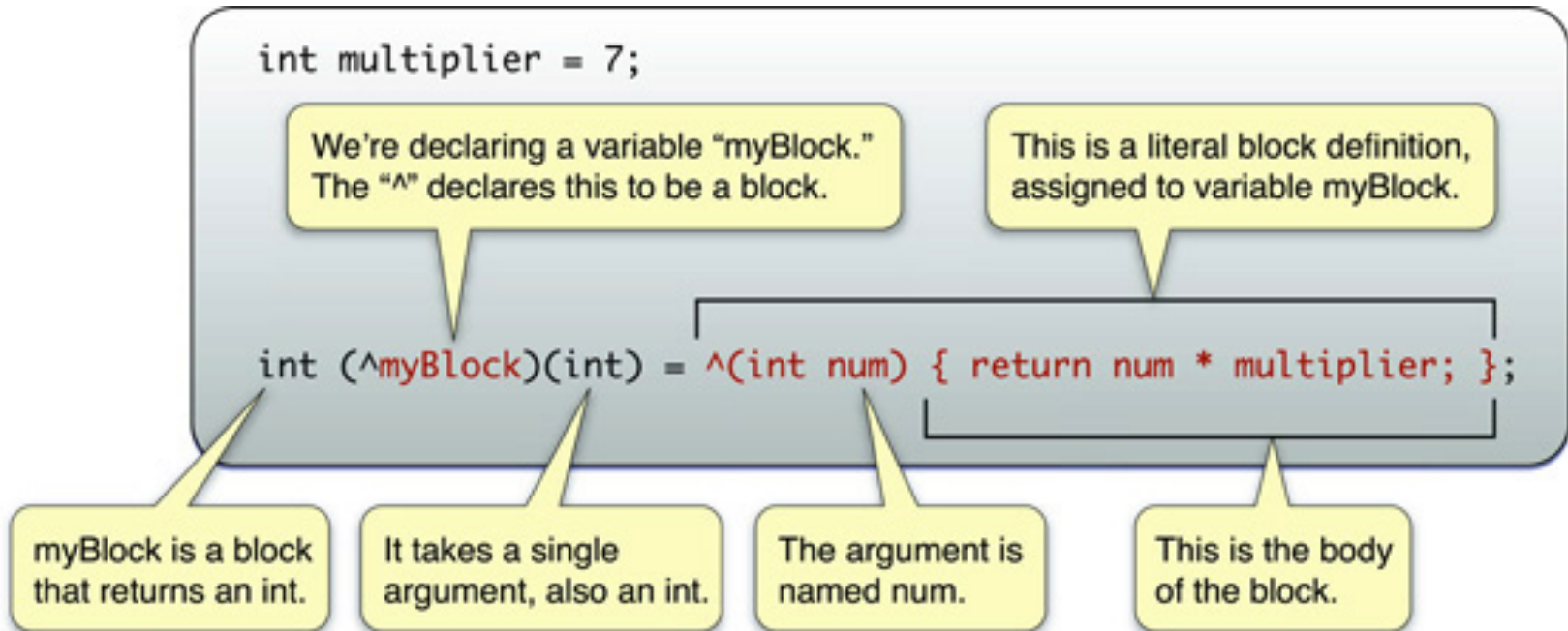
Source: iOS Documentation, Apple

# Motion Sensors (since iOS 4.0)

- Core Motion Framework
- CMMotionManager provides access to
  - raw accelerometer data (accelerometer)
  - raw rotation-rate data (gyroscope)
  - processed device-motion data (fused sensor data)
- Receiving sensor data
  - Specific intervals
    - accelerometerUpdateInterval
    - startAccelerometerUpdatesToQueue:withHandler:
  - On demand
    - startAccelerometerUpdates
    - accelerometerData

# Code Blocks

- Blocks: conceptually similar to function pointers in C
  - Can access variables from function in which they are defined
- Passing a code block as argument



Source: iOS Documentation, Apple

- Calling: `[someObject someFunction:myBlock];`

# Getting Accelerometer Values

- Create and configure motion manager

```
motionManager = [[CMMotionManager alloc] init];
motionManager.accelerometerUpdateInterval = 1.0; // sec
```

- Register handler and start updates

```
void (^accHandler)(CMAccelerometerData*, NSError*) =
 ^(CMAccelerometerData *accData, NSError *error)
 {
 NSLog(@"%f, %f, %f", accData.acceleration.x,
 accData.acceleration.y, accData.acceleration.z);
 };
```

```
[motionManager startAccelerometerUpdatesToQueue:
 [NSOperationQueue mainQueue] withHandler:accHandler];
```

- Stop updates

```
[motionManager stopAccelerometerUpdates];
```

should not use main queue



# Getting Accelerometer Values

- Create new queue

```
queue = [[NSOperationQueue alloc] init];
```

- Register handler and start updates

```
void (^accHandler)(CMAccelerometerData*, NSError*) =
 ^(CMAccelerometerData *accData, NSError *error)
 {
 NSLog(@"%f, %f, %f", accData.acceleration.x,
 accData.acceleration.y, accData.acceleration.z);
 };
```

```
[motionManager startAccelerometerUpdatesToQueue:queue
 withHandler:accHandler];
```

- Need to use main queue to update UI

# Update UI on Main Queue

```
void (^accHandler)(CMAccelerometerData*, NSError*) =

^(CMAccelerometerData *accData, NSError *error)
{
 dispatch_async(dispatch_get_main_queue(), ^{
 NSString *s;
 s = [NSString stringWithFormat:@"%%.3f", accData.acceleration.x];
 labelX.text = s;
 s = [NSString stringWithFormat:@"%%.3f", accData.acceleration.y];
 labelY.text = s;
 s = [NSString stringWithFormat:@"%%.3f", accData.acceleration.z];
 labelZ.text = s;
 });
};
```

# Operation Queues

- NSOperationQueue objects store and execute NSOperation objects
- Operation queues
  - Store and execute operation objects
  - Provide their own threads to execute operations
  - Encapsulate threads
- Operations
  - Abstract base classes for tasks to perform
  - Have priorities
  - One operation may depend on completion of other operations

# NETWORKING

# Synchronous Download of Data

- Waits until data completely downloaded

```
- (UIImage*) imageFromUrlString:(NSString*)urlString;
{
 NSURL *url = [NSURL URLWithString:urlString];
 NSData *data = [NSData dataWithContentsOfURL:url];
 UIImage *image = [UIImage initWithData:data];
 return image;
}
```

- Variant: specify options (e.g. caching) and error variable

```
NSDataReadingOptions o = NSDataReadingUncached;
NSError *e = nil;
NSData *data = [NSData dataWithContentsOfURL:url options:o error:&e];
UIImage *image = [UIImage initWithData:data];
if (e) {...}
```

# Synchronous Downloads with NSURLConnection

- Synchronous request with error response

```
NSMutableURLRequest *req = [NSMutableURLRequest
 requestWithURL:url];
```

```
NSURLResponse *res = nil;
```

```
NSError *e = nil;
```

```
NSData *data = [NSURLConnection sendSynchronousRequest:req
 returningResponse:&res error:&e];
```

- NSURLResponse contains content length, MIME type, etc.

# Asynchronous Downloads with NSURLConnection

- Put request on a second thread

```
[NSThread detachNewThreadSelector:@selector(downloadImage:)
toTarget:self withObject:urlString];
```

- Update GUI on main (GUI-safe) thread

```
[imageView performSelectorOnMainThread:@selector(setImage:)
withObject:image waitUntilDone:NO];
```

# Exercise 2: Multi Tab Application

