

Apple's new Swift language

Peter Himschoot
peter@u2u.be



Agenda

- Developing for iOS overview
 - Xcode, Objective-C, Swift, Development life cycle
- UI development
 - Interface Builder
 - Storyboard and segues
 - Universal layout
- Swift
 - Features
- Some APIs
 - Core Data, Services, Camera
 - MapKit Points of Interest and Directions
 - TouchID

Developing for iOS

u2u

- Only one active application
 - No real multi-tasking
 - Applications freeze when no longer active
 - Background processing possible
 - With limits
- Only one window
- Limited Access
 - Sandbox - store documents, preferences, data
 - Networking limitations, no low-level access
 - No root access

iOS specifics

u2u

- Limited Response Time
 - Snappy - fast - load fast, close fast
 - In case of incoming call - 5 seconds max
- Limited Screen Size
- Limited System Resources
 - 1 Gb RAM ~half for application
 - No swapping
- No Garbage collection
 - Automatic Reference Counting
- Battery consumption
 - “Battery shaming”



Programming Languages



- Objective-C
 - Object-Oriented adaptation of C, based on SmallTalk
 - Old, from the 80's
 - Verbose syntax
 - Supports dynamic programming
 - Apple added a lot of improvements, "Modern Objective-C"
- Swift
 - Modern Object-Oriented programming language
 - Similar to Java and C#
 - Interoperable with Objective-C
 - Brand new, announced in 2014

Microsoft and Apple Training



Demo

Building Hello world with Xcode
and Interface Builder

UI Development



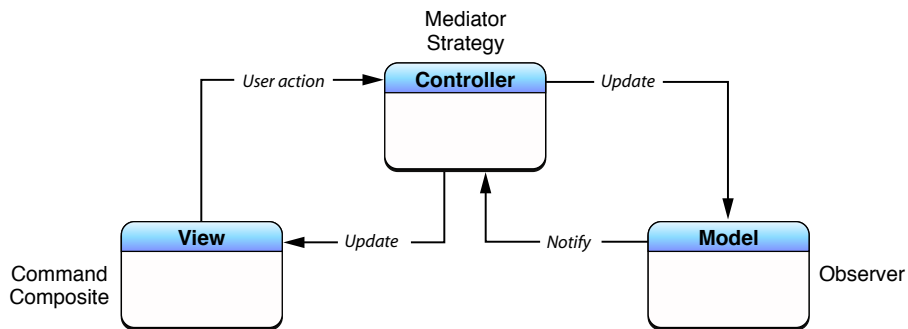
- Interface Builder
- Segues
- M-V-C
- Auto layout

Interface Builder



- Interface builder is the primary designer in XCode
 - Used to be separate tool
- You use it to build your views
 - And with a storyboard your navigation between views
- Interface builder uses **.xib** (also know as .nib) files
 - NextStep Interface **B**uilder
- And also uses **Storyboard** files
 - Think of this as collection of .xib files
 - With navigation through segues

Cocoa MVC design pattern

u2u

▪ source: Cocoa Design Patterns

Microsoft and Apple Training

u2u

Demo

Navigation with storyboards
and segues

Auto Layout





u2u

- Constraint based layout system
- Designing with intent
 - “The two buttons should have the same width”
 - “These textboxes should be aligned at the left side”
 - “This textbox should be to the right of its label”
- Define the constraints
 - The system does the layout

“Prepare for the future”

Unified Storyboard and Size Class

u2u

	Regular	Compact
Regular	 iPad Portrait Landscape	 iPhone Portrait
Compact	 ?	 iPhone Landscape

Interface Builder and Size class

u2u

- Interface Builder now allows you to select the size class
 - wAny – hAny means base layout
 - Install and uninstall constraints

Equal Widths Constraint

First Item: Cloud.Width

Relation: Less Than or Equal

Second Item: Superview.Width

+ Constant: 0

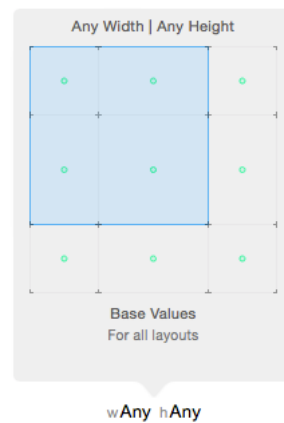
Priority: 1000

Multiplier: 0.45

Placeholder: ☐ Remove at build time

+ ☐ Installed

× wAny hC ☒ Installed



Microsoft and Apple Training

u2u

Demo

Building a weather app
with Xcode 6, a universal storyboard
and size classes

Swift – some features



- Swift vs Objective-C
- Tuples and Enumerations
- Optionals (!)
- Functions
- Closures

Swift, Apple's new language



- Modern Object Oriented programming language
- Full interoperability with Objective-C
- Type safe
- Type inference
- No **nil** valued variables
 - Optional types
- Functions are first class citizens
 - Closures
- Extensions

Programming languages

u2u

Objective-C

Rectangle.h

```
#import <Foundation/Foundation.h>
#import <UIKit/UIKit.h>

@interface Rectangle : NSObject

@property (assign, nonatomic) CGPoint * topleft;
@property (assign, nonatomic) CGSize * size;

- (Rectangle*) overlap: (Rectangle*) other;

@end
```

Rectangle.m

```
#import "Rectangle.h"

@implementation Rectangle

- (Rectangle *)overlap:(Rectangle *)other
{
    return nil;
}

@end
```

Swift

Rectangle.swift

```
import Foundation
import UIKit

class Rectangle {
    var topleft : CGRect?
    var size : CGSize?

    func overlap( other: Rectangle )
        -> Rectangle?
    {
        return nil
    }
}
```

Swift and Objective-C together

u2u

- Swift and Objective-C can be used in the same project
 - Uses bridging headers, normally created by Xcode



Would you like to configure an Objective-C bridging header?

Adding this file to TrySwift will create a mixed Swift and Objective-C target. Would you like Xcode to automatically configure a bridging header to enable classes to be accessed by both languages?

Cancel

No

Yes

Tuples



- Built in type

```
let notFound = (404, "Not Found")
let point = (10, 20, 30)
```

- Can be named

```
let named = (code: 404, text: "Not Found")
typealias httpError = (code: Int, text: String)
var error : httpError = notFound
```

- Accessing members

```
println("Error code \(${notFound.0}) means \(${notFound.1}")
println(error.code)
```

```
let (code, msg) = notFound
println("Error code \(${code}) means \(${msg}")
```

```
let (_, msg) = notFound // ignore code
```

Swapping variables



- Swap the value of two variables ?

- No need for a temporary variable

```
var a = 5
var b = 6
(a,b) = (b,a)
```

Enumerations

u2u

- Create a type-safe group of values

```
enum Direction {
    case North
    case East
    case South
    case West
}
```

- Shorter

```
enum Direction {
    case North, East, South, West
}
```

- Unlike most other languages not linked to integer type

Associated values

u2u

- Representing bar codes



- Using enumeration type

```
enum Barcode {
    case UPCA(Int, Int, Int, Int)
    case QRCode(String)
}
```

```
var productBarcode = Barcode.UPCA(8, 85909, 51226, 3)
```

Optionals



- Normal variables cannot be nil

```
var x : String = nil // compiler error
```

- Optionals are variables that may or may not have a value

```
var x : String? = ...
```

Unwrapping optionals



- To use the value of an optional we need to unwrap it

```
var i : Int?  
  
if i != nil { // test  
    let _i = i! // unwrap  
    println(_i)  
}
```

- The test and unwrap can be done as a single statement
 - Optional binding

```
if let _i = i {  
    println(_i)  
}
```

Optional chaining

u2u

- Consider following classes

```
class Address {
    var street: String?
    var city: String?
}

class Employee {
    var name: String?
    var address: Address?
}

class Department {
    var name: String?
    var manager: Employee?
}
```

- You want to get a department's manager's city

```
var d = Department()
```

- Would you like to write this?

```
if let _manager = d.manager {
    if let _address = _manager.address {
        if let _city = _address.city {
            println(_city.uppercaseString)
        }
    }
}
```

Optional Chaining

u2u

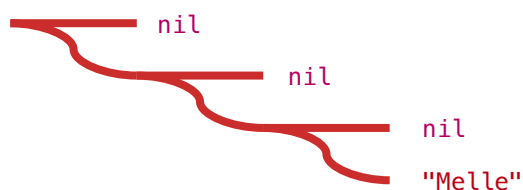
- Optional chaining allows you to write it like this

```
if let _city = d.manager?.address?.city?.uppercaseString {
    println(_city)
}
```

- Optional chaining follows the track of values

– Also works with methods

```
d.manager?.address?.city?.uppercaseString
```



Functions



- From Apple documentation

Functions are self-contained chunks of code that perform a specific task. You give a function a name that identifies what it does, and this name is used to “call” the function to perform its task when needed

- Declaring functions

- Named functions

```
func globalFunc(a: Int, b: Int) -> Int {
    return a+b
}
```

- Closures

Function variables



- Functions are types
- Can be assigned to variables and properties

```
var f = globalFunc
f(1,2)
```

```
f = obj.memberFunc
f(1,2)
```

```
var f2 : (Int,Int)->Int
f2 = globalFunc
f2 = obj.memberFunc
```


Functions are first class citizens



- Functions can return other functions

```
func greeterFunc(lang: String) -> (() -> String) {
    func dutch() -> String { return "Goede morgen" }
    func french() -> String { return "Bonjour" }

    if lang == "nl" {
        return dutch
    } else {
        return french
    }
}
```

- Storing and calling the result

```
let greeter = greeterFunc("nl")
greeter()
```

External Parameter names



- Consider following function call

```
vehicleFromStorage(false)
```

- Question: what does the boolean mean?

- Same function call:

```
vehicleFromStorage(ignoreCase: false)
```

- Same question: what does the boolean mean?

External Parameter names

u2u

- Apple recommendation

*Consider using external parameter names
whenever the purpose of a function's
arguments would be unclear to **someone**
reading your code for the first time*

You!

External Parameter names

u2u

- Declaring the function

```
func vehicleFromStorage(ignoreCase: Bool) -> Vehicle {
  ...
}
```

- How do you require the argument name?

– Use external parameter name before internal parameter name

```
func vehicleFromStorage(ignoreCase ic: Bool) -> Vehicle {
  ...
}
```

- Shorter, same internal as external parameter name

```
func vehicleFromStorage( #ignoreCase: Bool) -> Vehicle {
  ...
}
```

No external parameters names



- By default instance methods have external names
 - All but first parameter

```
func fuel(fuelType: String, liters: Int)
```

- Needs to be called like this:

```
v.fuel("Gas", liters: 50)
```

- You can turn this off by using `_` as parameter name

```
func fuel(fuelType: String, _ liters: Int)
```

- Now use:

```
v.fuel("Gas", 50)
```

Closures – Anonymous functions



- “Instruction” objects (aka Blocks / Lambda functions)
- Let's say we have an array of numbers

```
let numbers = [1, 2, 3, 4, 5]
```

- We want to calculate the average of a function

```
func square( a: Int ) -> Int {
  return a*a;
}
```

```
func averageOfSquares( #nrs: [Int] ) -> Double {
  var sum = 0;
  for nr in nrs {
    sum += square(nr)
  }
  return Double(sum) / Double(nrs.count)
}
```

Closures



- After a while we need the average of another function
 - Should we Copy – Paste ?

- No, we use a closure to pass the function

```
func averageOfFunction( #nrs : [Int], f: (Int -> Int))
  -> Double {
  var sum = 0;
  for nr in nrs {
    sum += f(nr)
  }
  return Double(sum) / Double(nrs.count)
}
```

- The argument f is a Function Type

Passing closures to functions



- Passing a compatible function

```
averageOfFunction(nrs: numbers, f: square)
```

- Verbose “closure expressions”

```
averageOfFunction(nrs: numbers,
  f: { (a: Int) -> Int in return a*a })
```

- Shorter, using type inference

```
averageOfFunction(nrs: numbers, f: { a in a*a })
```

- Even shorter, using default parameter name \$0

```
averageOfFunction(nrs: numbers, f: { $0 * $0 })
```

Trailing closure syntax



- Swift closures have a convenient syntax to pass closures
 - When the last argument is a closure

```
averageOfFunction(nrs: numbers) { $0*$0 }
```

- This is especially convenient with completion handlers!

```
let snapshotter = MKMapSnapshotter(options:options)
snapshotter.startWithCompletionHandler
{
    [unowned self] snapshot, error in
    if let _snapshot = snapshot {
        ...
    }
}
```

Some important APIs



- Core data
- AFNetworking
- Camera
- UIAlertController
- MapKit
 - Points of interest
 - Directions
- TouchID

What is Core Data?

u2u

- Core Data is an **Object Graph**
 - That **can** persist to a database
- Core Data manages the Object Graph
 - Persistence is secondary (but still very important)

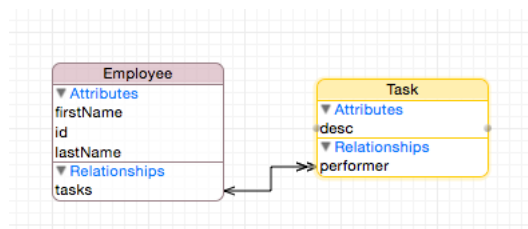


- Object Graph
 - Stored as .xcdatamodeld file in Xcode
 - Edited in Xcode

Advantages of Core Data

u2u

- Build a data model visually
 - define entities
 - define attributes – relationships – fetched properties
- Forget about persistence
- Uses SQLite as backing store
 - By default, other storage mechanisms are possible



Networking



- iOS has a rich open source community
 - Many have built extensions to use REST services
 - AFNetworking

```

NSURLRequest *request = [NSURLRequest requestWithURL:url];

AFJSONRequestOperation *operation =
    [AFJSONRequestOperation JSONRequestOperationWithRequest:request
     success:^(NSURLRequest *request, NSHTTPURLResponse *response, id JSON)
     {
         self.weather = (NSDictionary *)JSON;
         self.title = @"JSON Retrieved";
         [self.tableView reloadData];
     }
     failure:^(NSURLRequest *request, NSHTTPURLResponse *response, NSError *error)
     {
         ""
     }];

[operation start];

```

Camera



```

- (IBAction)takePictureTapped:(id)sender {

    self.imagePicker = [[UIImagePickerController alloc] init];
    self.imagePicker.sourceType = UIImagePickerControllerSourceTypeCamera;
    self.imagePicker.mediaTypes = [UIImagePickerController
        availableMediaTypesForSourceType:UIImagePickerControllerSourceTypeCamera];
    self.imagePicker.allowsEditing = YES;
    self.imagePicker.delegate = self;

    [self presentViewController:self.imagePicker animated:YES completion:nil];
}

- (void)imagePickerController:(UIImagePickerController *)picker
didFinishPickingMediaWithInfo:(NSDictionary *)info {

    // First dismiss the image picker
    [self.imagePicker dismissViewControllerAnimated:YES completion:^ {

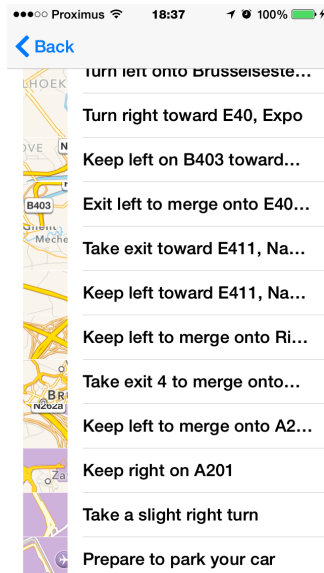
        UIImage * originalImage = info[UIImagePickerControllerOriginalImage];
        UIImage * editedImage = info[UIImagePickerControllerEditedImage];
        UIImage * pillImage = originalImage ? originalImage : editedImage;
        UIImage * pillThumbImage = [pillImage thumbnailImage:100
            transparentBorder: 0
            cornerRadius:10
            interpolationQuality:kCGInterpolationDefault];
        self.pillImageView.image = pillThumbImage;
    }];
}

```

From A to B using MapKit



- Use MapView to show map
- Use Points of interest
 - To find destination
- Use Directions
 - To find directions from and to airport
- Add MKAnnotations
 - To put route on map
- Step by step instructions



MapKit Points of Interest



Swift

```
var searchRequest = MKLocalSearchRequest()
searchRequest.naturalLanguageQuery = searchBar.text
var localSearch = MKLocalSearch(request: searchRequest)
localSearch.startWithCompletionHandler()
{ response, error in
    if let _response = response
    {
        ...
    }
}
```


MapKit Directions



Swift

```
var directionsRequest = MKDirectionsRequest()
directionsRequest.setSource(from)
directionsRequest.setDestination( to)
directionsRequest.transportType =
MKDirectionsTransportType.Automobile

var directions = MKDirections(request: directionsRequest)
directions.calculateDirectionsWithCompletionHandler
{
    (response: MKDirectionsResponse?, error: NSError?) in
    if let _count = response?.routes?.count
    {
        ...
    }
}
```

Microsoft and Apple Training



Demo

Using Core Data