the

iOS 7 DEV CRASH COURSE

By Nicolas Goles

CONTENTS

- Objective-C
- MVC
- Delegation & Blocks
- HTTP
- Persistence
- Geolocation
- Xcode
- Interface Builder

what's

OBJECTIVE-C

OBJECTIVE-C

- Superset of the C programming language.
- Adds syntax for object-oriented capabilities.
- Provides dynamic typing and binding.
- Defers many responsibilities until runtime.

* Objective-C Intro

And :)

"In Objective-C one does not simply call a method; one sends a message..."

POP QUIZ

- How to log a message to the console?
- How do we call a method in Objective-C?
- How do we declare a Class in Objective-C?

console log:

```
NSLog(@“Is there anybody out there?”);
```

method call:

```
[NSLayoutConstraint constraintWithItem:view1
                             attribute:NSLayoutAttributeHeight
                             relatedBy:NSLayoutRelationEqual
                                toItem:view2
                             attribute: NSLayoutAttributeHeight
                            multiplier:0.5
                              constant:0];
```

Class

```
//  Often in Foo.h
@interface Foo : NSObject

@end

//  Often in Foo.m
@implementation Foo

@end
```

what's

COCOA TOUCH

Cocoa Touch

Complete Assortment of Frameworks for building iOS Applications.

* Cocoa Touch Layer

Audio & Video

Data Management

Graphics

User Applications

Networking

Why

MVC

MVC

Basic Design Pattern to build an iOS App.

MVC

let's start...

BLOCKS & DELEGATES

DELEGATION

- Base communication pattern used by the iOS APIs.

- Customize an object's behavior and be notified about certain events.

eg:

```
// Foo.h
@interface Foo : UITableViewController <UIAlertViewDelegate>

@end

// Foo.m
@implementation Foo
- (void)showAlert {
 UIAlertView *alert = [[UIAlertView alloc] initWithTitle:@"Test Alert"
                                                 message:@"EHLO :)"
                                                delegate:self
                                       cancelButtonTitle:@"OK"
                                       otherButtonTitles:nil];
    [alert show];
}
@end
```

eg:

```
// Foo.m
//

@implementation Foo

- (void)showAlert {
  UIAlertView *alert = [[UIAlertView alloc] initWithTitle:@"Test Alert"
                                                  message:@"EHLO :)"
                                                 delegate:self
                                        cancelButtonTitle:@"OK"
                                        otherButtonTitles:nil];
    [alert show]; // Will show Alert View
}

// Delegate Method
- (void)alertView:(UIAlertView *)alertView clickedButtonAtIndex:(NSInteger)buttonIndex {
  NSLog(@"User pressed Alert’s View button # %d!”, buttonIndex);
}

@end
```

BLocks

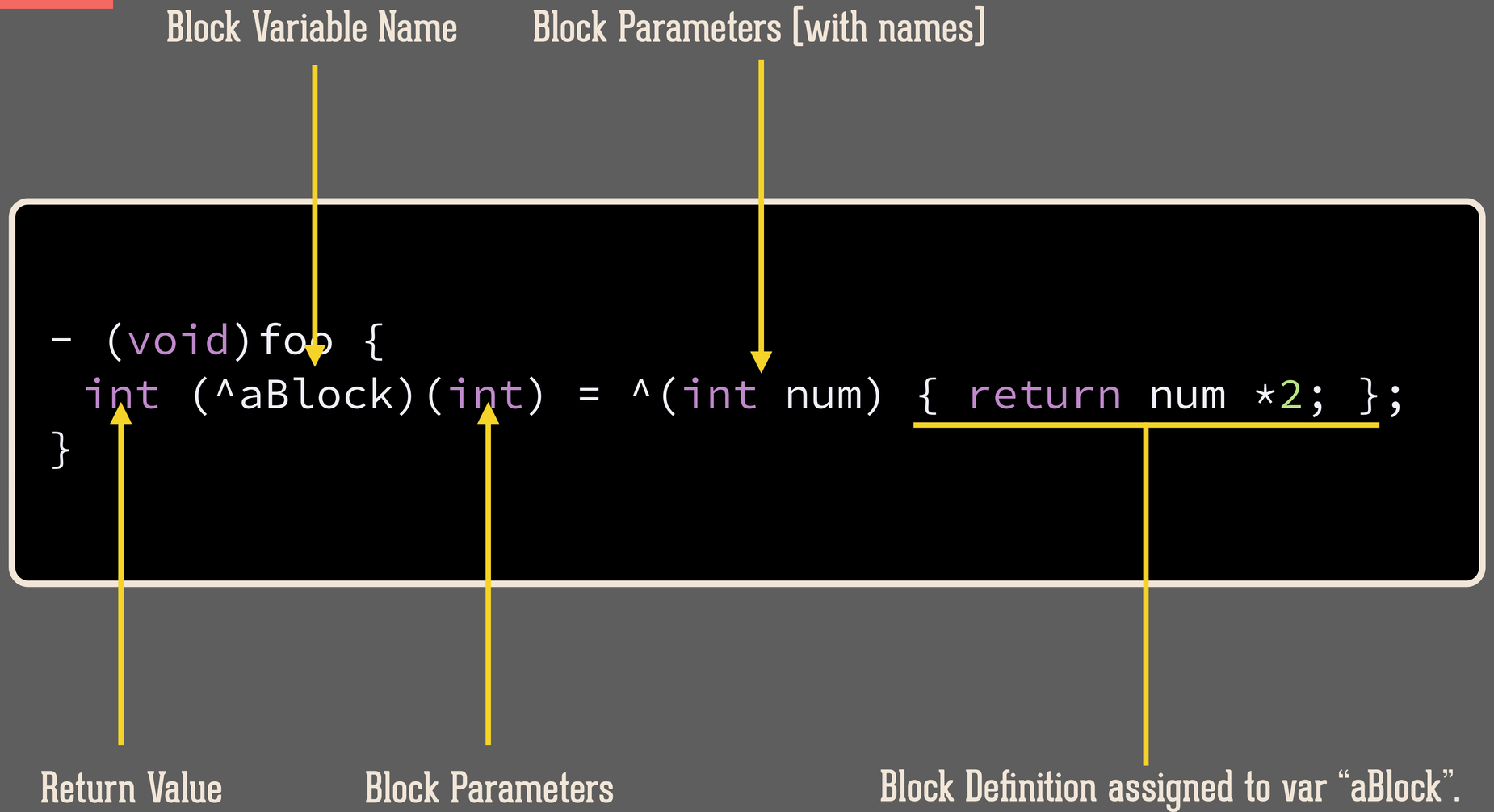
BLOCKS

- “New” to Objective-C*
- Can often do what could be implemented using delegation.**

* iOS 4+

** Both delegation & blocks have their advantages & requirements.

BLOCKS


Block Variable Name
Block Parameters [with names]
- (void)foo {
int (^aBlock)(int) = ^(int num) { return num *2; };
}
Return Value
Block Parameters
Block Definition assigned to var “aBlock”.

BLOCKS

eg:

```
- (void)foo {
  int (^myBlock)(int) = ^(int num) { return num * 2; };
  int four = myBlock(2); // 2*2 == 4
}
```

how to...

HTTP

NSURLSession

The NSURLSession class and related classes provide an API for downloading content via HTTP.*

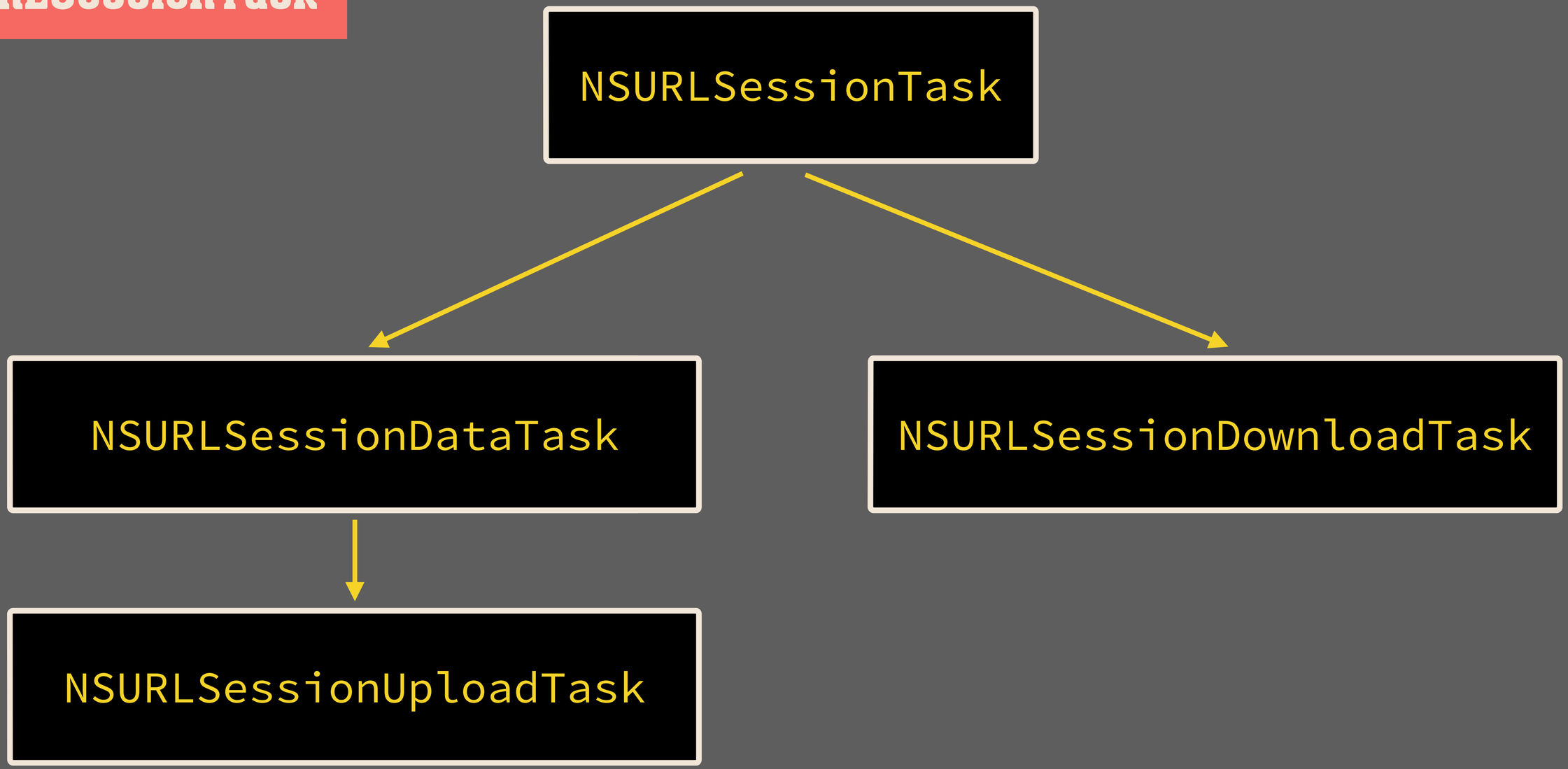
* iOS 7+

NSURLSession

config

```
NSURLSessionConfiguration *config = [NSURLSessionConfiguration defaultSessionConfiguration];
config.HTTPAdditionalHeaders = @{@"Accept" : @"text/json"};
_session = [NSURLSession sessionWithConfiguration:config delegate:nil delegateQueue:nil];
```

NSURLSessionTask



NSURLSessionTask

GET

```
NSURL *URL = [NSURL URLWithString:@"http://example.com"];
NSURLRequest *request = [NSURLRequest requestWithURL:URL];

NSURLSession *session = [NSURLSession sharedSession];
NSURLSessionDataTask *task = [session dataTaskWithRequest:request
                                        completionHandler:
                              ^(NSData *data, NSURLResponse *response, NSError *error) {
                                  // ...
                              }];

[task resume];
```

NSURLSessionTask

JSON

```
NSURL *URL = [NSURL URLWithString:@"http://example.com/json_service"];
NSURLRequest *request = [NSURLRequest requestWithURL:URL];

NSURLSession *session = [NSURLSession sharedSession];
__block NSDictionary *JSON;
NSURLSessionDataTask *task = [session dataTaskWithRequest:request
                                        completionHandler: ^(NSData *data, NSURLResponse *response, NSError *error) {
    //Remember to Check for HTTP errors or NSError

    NSError *parsingError;
    JSON = [NSJSONSerialization JSONObjectWithData:data options:0 error:&parsingError];
    if (!JSON) {
        NSLog(@"Handle Error %@", [parsingError localizedDescription]);
}
    NSLog(@"%@", JSON[@"Some Key"]);
}];

[task resume];
```

how to...

PERSIST DATA

NSKeyedArchiver

Entity Modeling: NO

Querying: NO

Speed: Slow*

Serialization Format: NSData

Migrations: Manual

* Speed is somewhat relative... but no, NSKeyedArchiver is **not** the Ferrari of Serialization.

NSKeyedArchiver

Does the Job: YES

Painful to Use: NO

* http://nshipster.com/nscoding/

NSKeyedArchiver

Simple 2 method @protocol

```
-initWithCoder:
encodeWithCoder:
```

NSCoding

```objc
@interface Book : NSObject <NSCoding>
@property NSString *title;
@property NSUInteger pageCount;
@property NSSet *categories;
@end

@implementation Book

// NSCoding
- (instancetype)initWithCoder:(NSCoder *)decoder {
   self = [super init];
   if (!self) {
      return nil;
    }

   self.title = [decoder decodeObjectForKey:@"title"];
   self.pageCount = [decoder decodeIntegerForKey:@"pageCount"];
   self.categories = [decoder decodeObjectForKey:@"categories"];

   return self;
}

- (void)encodeWithCoder:(NSCoder *)encoder {
   [encoder encodeObject:self.title forKey:@"title"];
   [encoder encodeInteger:self.pageCount forKey:@"pageCount"];
   [encoder encodeObject:self.categories forKey:@"categories"];
}

@end
```

NSCoding

```
- (void)storeBook {
  Book *book = [[Book alloc] init];
  book.title = @"The man in the High Castle";
  book.pageCount = 288;
  book.categories = [[NSSet alloc] initWithArray:@[@"Science Fiction", @"Alternate History"]];

  NSData *data = [NSKeyedArchiver archivedDataWithRootObject:book];
  [[NSUserDefaults standardUserDefaults] setObject:data forKey:@"theBook"]; // DON'T DO THIS!!
}
```

where??

GEO LOCATION

CLLocationManager

"The CLLocationManager class defines the interface for configuring the delivery of location- and heading-related events to your application."*

* CCLocationManager Reference

CLLocationManager

- Part of the CoreLocation Framework.
- Works with the delegation pattern.
- Will ask for user permission.

CLLocationManager

```objc
#import <CoreLocation/CoreLocation.h>

@interface FooLocation : AController <CLLocationManagerDelegate>
@property (nonatomic, strong) CLLocationManager *locationManager;
@end

@implementation FooLocation

- (instancetype)init {
  if (self = [super init]) {
   _locationManager = [[CLLocationManager alloc] init];
   _locationManager.delegate = self;
   _locationManager.distanceFilter = kCLDistanceFilterNone;
   _locationManager.desiredAccuracy = kCLLocationAccuracyBest;
   [_locationManager startUpdatingLocation];
   }
   return self;
}

// Delegate Implementation
- (void)locationManager:(CLLocationManager *)manager didUpdateLocations:(NSArray *)locations {
  CLLocation *lastLocation = [locations lastObject];
  CLLocationCoordinate2D lastCoordinate = lastLocation.coordinate;
  NSLog(@"Last [lat, long] - [%f, %f]", lastCoordinate.latitude, lastCoordinate.longitude);
}

@end
```

Tools?

XCODE

Xcode Demo

&

The End :)