# iOS Development with Blocks

Jeff Kelley (@jeff_r_kelley)

MobiDevDay Detroit | February 19, 2011

Slides will be on blog.slaunchaman.com

# Introduction

- Blocks
  - What are blocks?
  - Why should we use blocks?
- Grand Central Dispatch
  - How does GCD help iOS development?

# Blocks

- Apple extension to the C language
- Similar to closures, lambdas, etc.
- Encapsulate code like a function, but with extra “magic”

# Your First Block

```
void (^helloWorldBlock)(void) = ^ {
    printf ("Hello, World!\n");
};
```

# Calling Your First Block

```
void (^helloWorldBlock)(void) = ^ {
    printf ("Hello, World!\n");
};

helloWorldBlock();
```

# Blocks That Return

- Blocks return just like functions.

```
int (^fortyTwo)(void) = ^{
    return 42;
}


int a = fortyTwo();
```

# Blocks That Take Arguments

- Just like functions:

```
BOOL (^isFortyTwo)(int) = ^(int a) {
    return (a == 42);
}

BOOL myBool = isFortyTwo(5); // NO
```

# Blocks Capture Scope

```
int a = 42;

void (^myBlock)(void) = ^{
    printf("a = %d\n", a);
};

myBlock();
```

```
int a = 42;

void (^myBlock)(void) = ^{
    a++;
    printf("a = %d\n", a);
};

myBlock();
```

```
__block int a = 42;

void (^myBlock)(void) = ^{
    a++;
    printf("a = %d\n", a);
};

myBlock();
```

# Block Typedefs

- Simple block that returns an int and takes a float:

```
typedef int (^floatToIntBlock)(float);
```

- Now initialize it:

```
floatToIntBlock foo = ^(float a) {
    return (int)a;
};
```

# Using Blocks as Arguments

- Method Signature:
- - (void)doThisBlock:(void (^)(id obj))block
- returns void, takes id argument

# Block Typedefs As Arguments

- First typedef the block, then use it as an argument:

```
typedef int (^floatToIntBlock)(float);

void useMyBlock(floatToIntBlock foo);
```

# Block Scope

```
void (^myBlock)(void);

if (someValue == YES) {

    myBlock = ^{ printf("YES!\n") };

} else {

    myBlock = ^ { printf("NO!\n") };

}
```

# Block Memory Management

- Block_copy() and Block_release()
- Copied onto heap
- Blocks are Objective-C objects!
  - [myBlock copy];
  - [myBlock release];

# Block Memory Management

- Unlike Objective-C objects, blocks are created on the stack
- Block_copy() moves them to the heap
- Block_retain() **does not**.
- Can also call [myBlock autorelease]

# Blocks Retain Objects

```
NSString *helloWorld = [[NSString alloc]
initWithString:@"Hello, World!"];

void (^helloBlock)(void) = ^{

  NSLog(@"%@", helloWorld);

};

[helloWorld release];

helloBlock();
```

# What Are Blocks Good For?

- Replacing Callbacks
- Notification Handlers
- Enumeration

# What Are Blocks Good For?

- Replacing Callbacks
- Notification Handlers
- Enumeration

# Replacing Callbacks

- Old UIView animations:

```
[UIView beginAnimations:@"foo" context:nil];

[UIView setAnimationDelegate:self];

[UIView setAnimationDidStopSelector:@selector
(animationDidStop:finished:context:)];
```

- Then you implement the callback

# Replacing Callbacks

```
- (void)animationDidStop:(NSString *)animationID
finished:(NSNumber *)finished context:(void *)context
{

    if ([animationID isEqualToString:kIDOne]) {

        ...

    } else if ([animationID isEqualToString:kIDTwo]) {

        ...

    }

}
```

# Replacing Callbacks

- New UIView Animations:

```
[UIView animateWithDuration:0.3
    animations:^{
        [myView setAlpha:0.0f];
    }
    completion:^(BOOL finished) {
        [self doSomething];
    }];
```

# Replacing Callbacks

- New UIView Animations:

```
[UIView animateWithDuration:0.3
        animations:^{
            [myView setAlpha:0.0f];
        }
         completion:^(BOOL finished) {
            [self doSomething];
        }];
```

# Replacing Callbacks

- New UIView Animations:

```
[UIView animateWithDuration:0.3
    animations:^{
        [myView setAlpha:0.0f];
    }
    completion:^(BOOL finished) {
        [self doSomething];
    }];
```

# Replacing Callbacks

- New UIView Animations:

```
[UIView animateWithDuration:0.3
    animations:^{
        [myView setAlpha:0.0f];
    }
    completion:^(BOOL finished) {
        [self doSomething];
    }];
```

# What Are Blocks Good For?

- Replacing Callbacks
- Notification Handlers
- Enumeration

# Notification Handlers

- Old Notifications:

```
[notificationCenter addObserver:self
                        selector:@selector(foo:)
                            name:kMyNotification
                          object:myObject];
```

# Notification Handlers

- Old Notifications: Userinfo Dictionaries

```
- (void)foo:(NSNotification *)aNotification
{
  NSDictionary *userInfo = [aNotification userInfo];
  NSString *UID = [userInfo objectForKey:kIDKey];
  NSNumber *value = [userInfo
objectForKey:kValueKey];
}
```

# Notification Handlers

- New Notification Handlers:

```
[notificationCenter addObserverForName:kName
                                object:myObject
                                 queue:nil
            usingBlock:^(NSNotification *aNotification) {
    [myObject doSomething];
}];

[myObject startLongTask];
```

# What Are Blocks Good For?

- Replacing Callbacks
- Notification Handlers
- Enumeration

# Enumeration

- Basic Enumeration

```
for (int i = 0; i < [myArray count]; i++)
{
    id obj = [myArray objectAtIndex:i];
    [obj doSomething];
}
```

# Enumeration

- But...

```
for (int i = 0; i < [myArray count]; i++) {
    id obj = [myArray objectAtIndex:i];
    [obj doSomething];

    for (int j = 0; j < [obj count]; j++) {
        // Very interesting code here.
    }
}
```

# Enumeration

- NSEnumerator

```
NSArray *anArray = // ... ;
NSEnumerator *enumerator = [anArray
objectEnumerator];

id object;

while ((object = [enumerator nextObject])) {

   // do something with object...

}
```

# Enumeration

- Fast Enumeration (10.5+)

```
for (id obj in myArray) {
    [obj doSomething];
}
```

# Enumeration

- Block-based enumeration

```
[myArray enumerateObjectsUsingBlock:
    ^(id obj, NSUInteger idx, BOOL *stop) {
        [obj doSomething];
    }];
```

# Enumeration

- So why use block-based enumeration?
  - -enumerateObjectsWithOptions:usingBlock:
    - NSEnumerationReverse
    - NSEnumerationConcurrent
  - Concurrent processing! And for free!

# What Are Blocks Good For?

- Replacing Callbacks
- Notification Handlers
- Enumeration

# What Are Blocks Good For?

- Replacing Callbacks
- Notification Handlers
- Enumeration
  - Sorting

# What Are Blocks Good For?

- Replacing Callbacks
- Notification Handlers
- Enumeration
  - Sorting

# Sorting

- NSArray
  - -sortedArrayWithOptions:usingComparator:
  - Concurrent sorting for free!

# Blocks Wrap-Up

- Carat Syntax
- Block Memory Management
- Using Blocks as Arguments
- Replacing Callbacks, Notification Handlers, Enumeration, and Sorting

# Intro to Grand Central Dispatch

# Moore's Law

- The quantity oftransistors that can be placed inexpensively on an integrated circuit has doubled approximately every two years. The trend has continued for more than half a century and is not expected to stop until 2015 or later.

Wikipedia

# Grand Central Dispatch (GCD)

- Open-source threading library (libdispatch)
  - Apache license
- Automatically optimizes threading
- Provides queues, timers, event handlers, etc.

# Simple GCD

```
dispatch_queue_t queue =
dispatch_get_global_queue
(DISPATCH_QUEUE_PRIORITY_HIGH,
0ul);

dispatch_async(queue, ^{

    [self performLongTask];

}
```

# Basic Dispatch Functions

- dispatch_async(queue, block);
  dispatch_async_f(queue, context, func);
  - Schedules block or function on queue, returns immediately
- dispatch_sync(queue, block);
  dispatch_sync_f(queue, context, func);
  - Schedules block or function on queue, blocks until completion

# Dispatch Queues

- dispatch_queue_t
- Main Queue
  - Analogous to main thread (Do your UI operations here)
  - dispatch_get_main_queue()

# Global Queues

- dispatch_get_global_queue(priority, flags);
  - priority is one of three constants:
    - DISPATCH_QUEUE_PRIORITY_LOW
    - DISPATCH_QUEUE_PRIORITY_NORMAL
    - DISPATCH_QUEUE_PRIORITY_HIGH
  - secong arg should always be 0ul (for now)

# Making Queues

- dispatch_queue_create(label, attr)
  - Use reverse DNS for label
    - com.example.myQueue
  - pass NULL for attr
- Be sure to use dispatch_release()

# Using Queues

- Custom queues are serial
  - First-in, first-out, one at a time
- Global queues are concurrent
  - GCD automatically chooses how many (usually # of CPU cores)