The background of the slide is a dark blue, textured surface resembling a spiral-bound notebook. A silver metal spiral binding is visible along the top edge, with the wire looping through a series of holes. The main title is centered on the page in a large, white, serif font.

# Concurrent Operations

The Very Best Way to Support Network Fetches



# Overview

- Brief NSOperation Review
- Apple's Advice
- Concurrent NSOperation Structure
- Points to keep in mind

# NSOperation

- ☐ Read the Concurrency Programming Guide
- ☐ Generally completes one task
- ☐ Setup the operation object, then enqueue
- ☐ Concurrent and Non-concurrent Styles

# Non-concurrent Operations

- ❑ Once started, runs to completion in its thread
- ❑ `NSInvocationOperation`
- ❑ `NSBlockOperation`
- ❑ `NSOperation` subclass



# Concurrent Operations

- ☐ Once started, runs asynchronously in its thread
- ☐ `NSOperation` subclass returning YES for `isConcurrent` subclass
- ☐ Given sparse treatment in Apple's Docs [rdar://9555852](http://rdar://9555852)
- ☐ "Most developers should never need to implement concurrent operation objects."



# Concurrent Testbed (iOS)

<https://public.me.com/dhoerl>

xcode folder

Concurrent\_NSOperation.zip

# Initial Screen

Carrier  8:13 AM 

Cancel Before Starting: ☐ OFF

[Start Op](#)

Fail during setup: ☐ OFF

[Message Op](#)

[Finish Op](#)

[Cancel Op](#)

[Run Connection](#)



# Interface

```
@interface ConcurrentOp : NSOperation
{
@property (assign) BOOL failInSetup;
@property (assign) NSThread *thread;
@property (retain) NSMutableData *webData;

- (void)wakeUp;
- (void)runConnection;

@end
```



# Implementation Part 1

```
@interface ConcurrentOp ()  
@property (nonatomic, assign) BOOL executing, finished;  
@property (nonatomic, assign) int loops;  
@property (nonatomic, retain) NSTimer *timer;  
@property (nonatomic, retain) NSURLConnection *connection;  
// @property (retain) id someResult; // I/O should be atomic
```

- (BOOL) setup;
- (void) finish;
- (void) timer: (NSTimer \*) timer;

```
@end
```

# Implementation Part 2

- (BOOL)isConcurrent { return YES; }
- (BOOL)isExecuting { return executing; }
- (BOOL)isFinished { return finished; }



# Implementation Part 3

```
- (void)start // NSOperation override
{
    if([self isCancelled]) { [self finish]; return; }

    self.thread = [NSThread currentThread];
    self.timer = [NSTimer scheduledTimerWithTimeInterval:60*60
        target:self selector:@selector(timer:) userInfo:nil
        repeats:NO];

    [self willChangeValueForKey:@"isExecuting"];
    executing = YES;
    [self didChangeValueForKey:@"isExecuting"];

    BOOL allOK = [self setup];
    ... /* Part 4 */
}
```

# Implementation Part 4

```
if(allOK){
    while(![self isFinished]) {
        BOOL ret = [[NSRunLoop currentRunLoop]
                    runMode:NSDefaultRunLoopMode
                    beforeDate:[NSDate distantFuture]];
        assert(ret);
    }
} else {
    [self finish];
}
```



# Implementation Part 5

```
- (BOOL) setup
{
    NSURLRequest *request = [NSURLRequest requestWithURL:
                             [NSURL URLWithString:
                              @"http://images.apple.com/home/images/icloud_title.png"]];

    self.connection = [[[NSURLConnection alloc]
                        initWithRequest:request
                        delegate:self
                        startImmediately:NO] autorelease];
    #if MAC // bug rdar://9621536
        [connection scheduleInRunLoop:[NSRunLoop currentRunLoop]
                        forMode:NSDefaultRunLoopMode];
    #endif
    return !failInSetup; // testing
}
```

# Implementation Part 6

```
- (void)wakeUp
{
    NSLog(@"WAKEUP!!!");
    if(loops++ >= 4)
        [self performSelector:@selector(finish) onThread:thread
withObject:nil waitUntilDone:NO];
}

- (void)runConnection
{
    [connection performSelector:@selector(start) onThread:thread
withObject:nil waitUntilDone:NO];
}
```



# Implementation Part 7

```
- (void)cancel
{
    [super cancel];

    if([self isExecuting]) {
        [self performSelector:@selector(finish) onThread:thread
withObject:nil waitUntilDone:NO];
    }
}
```

# Implementation Part 8

```
- (void) finish
{
    [self willChangeValueForKey:@"isFinished"];
    [self willChangeValueForKey:@"isExecuting"];

    executing = NO;
    finished = YES;

    [self didChangeValueForKey:@"isExecuting"];
    [self didChangeValueForKey:@"isFinished"];
}
- (void) dealloc
{
    [timer invalidate], [timer release];
    [connection cancel], [connection release];
    [webData release];

    [super dealloc];
}
```



# Running the Operation

```
- (IBAction)runNow:(id)sender
{
    ...
    [runner addObserver:self forKeyPath:@"isFinished" options:0
     context:runnerContext];

    // Have to be observing to get isFinished which happens when this
    // case below is hit
    if(preCancel.on) [runner cancel]; // Testing

    [queue addOperation:runner];
}
```

# Operation Finished

```
- (void)observeValueForKeyPath:(NSString *)keyPath...
{
    if(context == runnerContext) {
        if(runner.isFinished == YES) {
            // we get this on the operation's thread
            [self performSelectorOnMainThread:@selector
(operationDidFinish) withObject:nil waitUntilDone:NO];
        } else {
            NSLog(@"NSOperation starting to RUN!!!");
        }
    } else {
        [super observeValueForKeyPath:keyPath...;
    }
}
```



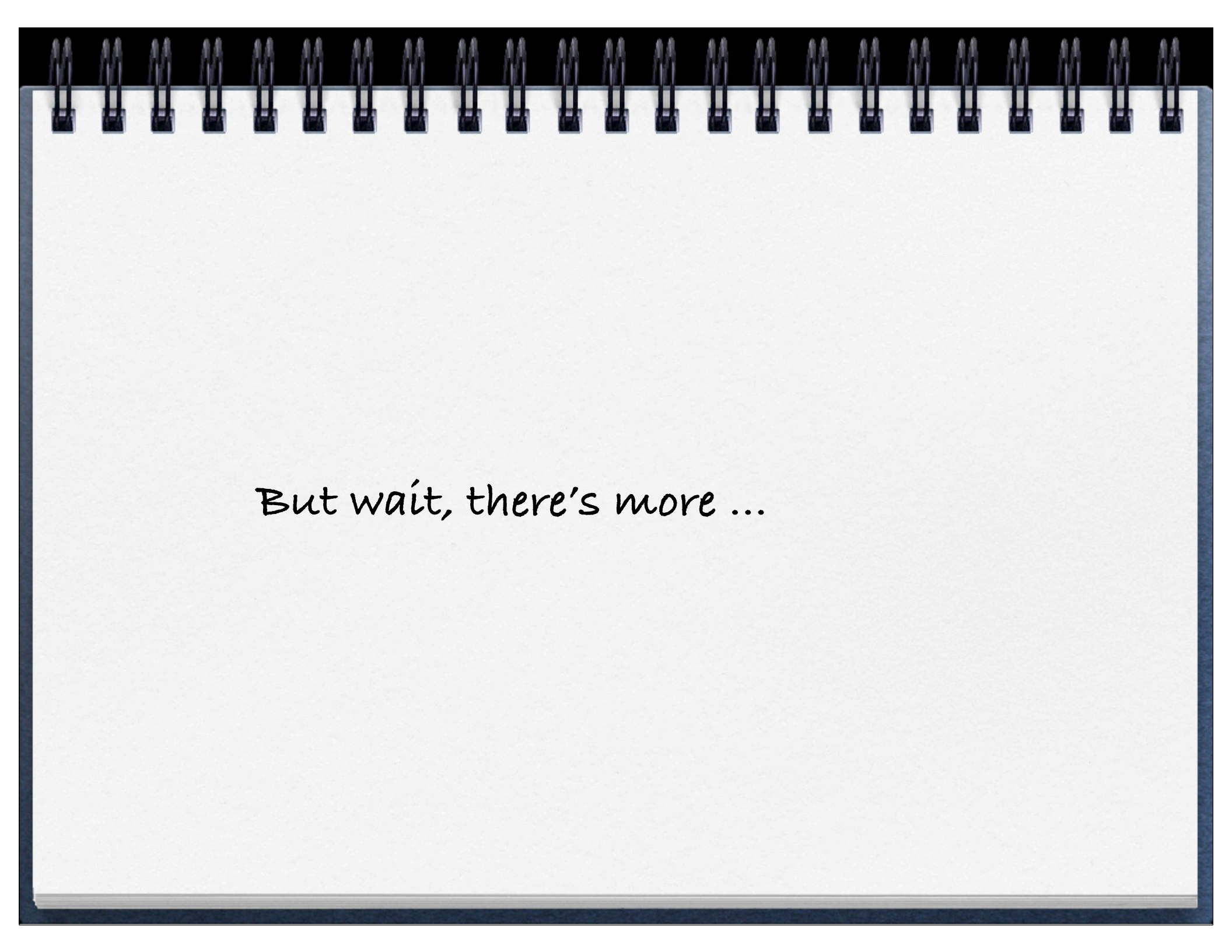
A top-down view of an open spiral-bound notebook. The notebook has a dark blue cover and a silver metal spiral binding along the top edge. The left page is blank white paper. The right page is also white but features the word "Demo..." in a large, bold, black sans-serif font, centered horizontally and positioned in the upper half of the page.

**Demo...**

# Concluding Remarks

- ❑ Operations run in a separate thread, so keep in mind when messaging it
- ❑ Threads can be cancelled before they run, so make sure you handle all edge cases.





But wait, there's more ...

# Breaking (public) News

- Automatic Reference Counting
- <http://clang.llvm.org/docs/AutomaticReferenceCounting.html>.