Deep Dive into (AF)Networking

App Subsystems

UI

Networking

Data Model

Logging

Image Processing

What we'll learn today

- Why AFNetworking?
- How to write an app using AFNetworking
- The Architecture of AFNetworking
- Problems with AFNetworking
- How to test your networking code

Why AFNetworking?

Different approaches

- Sockets
- NSURLConnection synch. requests + separate thread
- NSURLConnection asynchronous requests
- Third-party library (ASIHTTPRequest, etc...)

Requirements

- Need a way to perform several operations at the same time
- Need a way to have a retry mechanism configurable per operation
- Need a good way to define dependencies between operations
- Need to have one point of entry for API transactions so we can regulate network load and prioritize operations
- Need a way to use blocks for simplicity

Why AFNetworking?

- Eskimo at WWDC 2010 (Network Apps for iPhone OS (I and II), LinkedImageFetcher):

  Networking is hard

  If you want to do networking, do it asynchronously (so that you can cancel operations, main thread is not blocked etc...)

  Use NSURLConnection and NSOperationQueue

- Clean, easy-to-ready code and widely used nowadays
- ASIHTTPRequest is no longer supported the its author suggests to use AFNetworking as his first choice

How to write an app

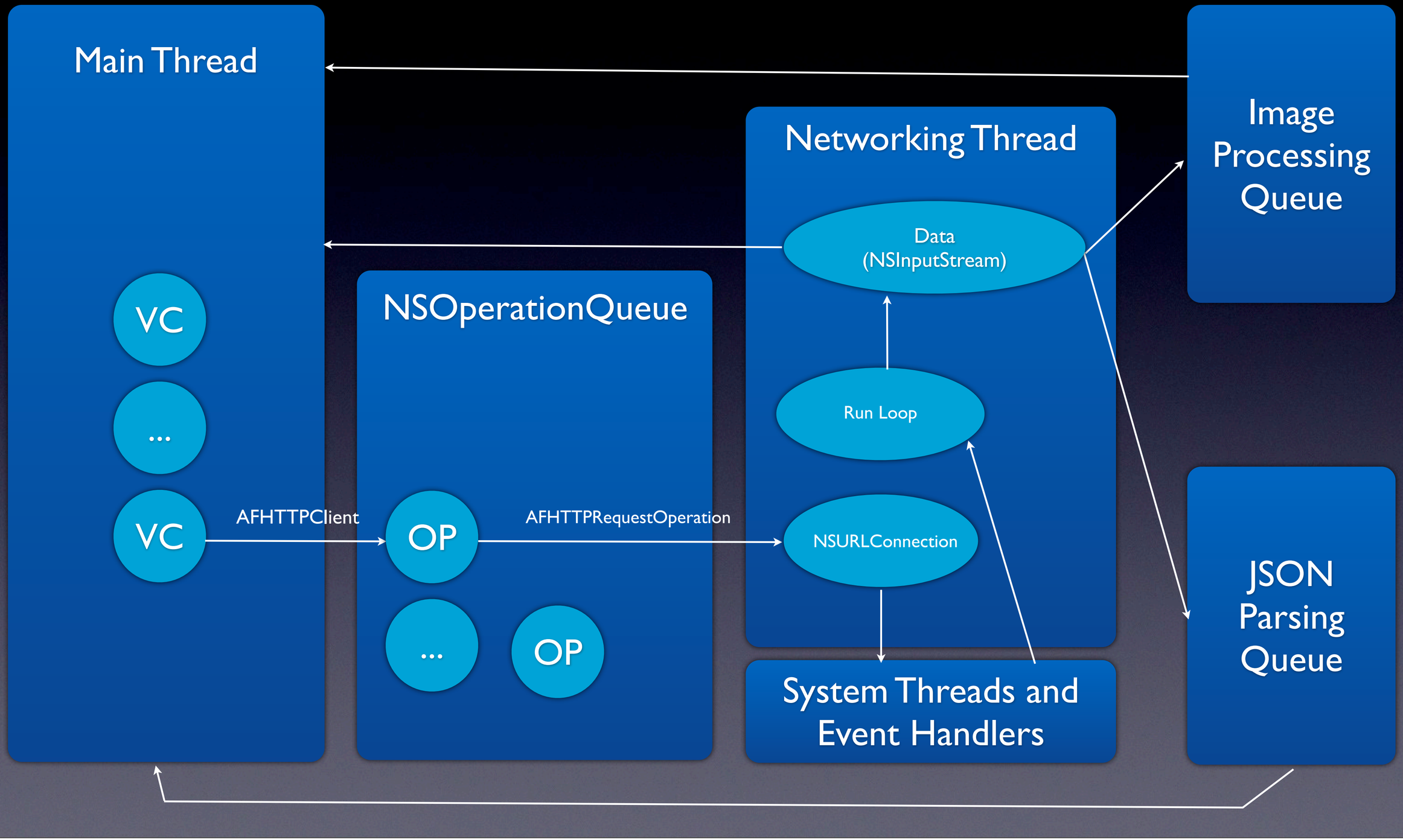
Shrek

Palo Alto

Source Code

- git clone git://github.com/andyegorov/shrek.git
- git checkout chapter_one - AFNetworking
- AFNetworking: https://github.com/AFNetworking/AFNetworking

The Architecture of AFNetworking



Operations

AFJSONRequestOperation (+XML, +plist)	JSON (+XML, +plist) parsing
AFHTTPRequestOperation	Acceptable error codes, content types HTTP error creation
AFURLConnectionRequestOperation	NSURLConnection creation Makes HTTPRequest and gets the response NSStream Management Progress block management

AFHTTPClient

- Keeps NSOperationQueue
- Manages batch operations and dispatch groups
- Base64 and URL encoding
- Construction of HTTP requests (HTTP headers, multipart form requests)

Groupon Additions to AFNetworking

- Retry and reauth mechanisms
- Reachability
- Operation groups for canceling

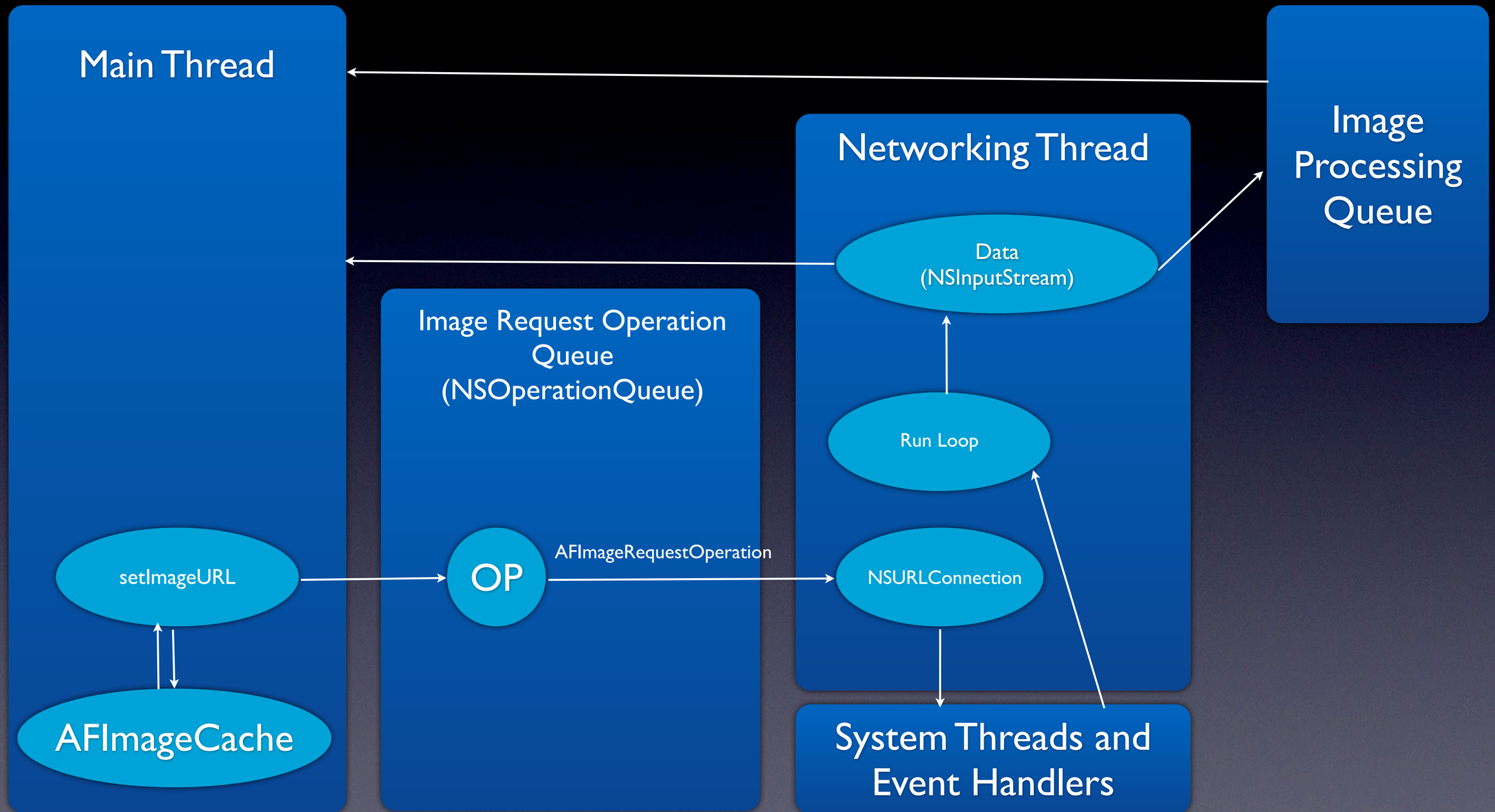
Operations

GrouponAPIJSONRequestOperation	JSON parsing
GrouponHTTPRequestOperation	Retry logic Operation groups for canceling Start and end time for NST tracking
AFHTTPRequestOperation	Dispatch groups Acceptable error codes, content types HTTP error creation
AFURLConnectionRequestOperation	NSURLConnection creation Makes HTTPRequest and gets the response NSStream Management Progress block management

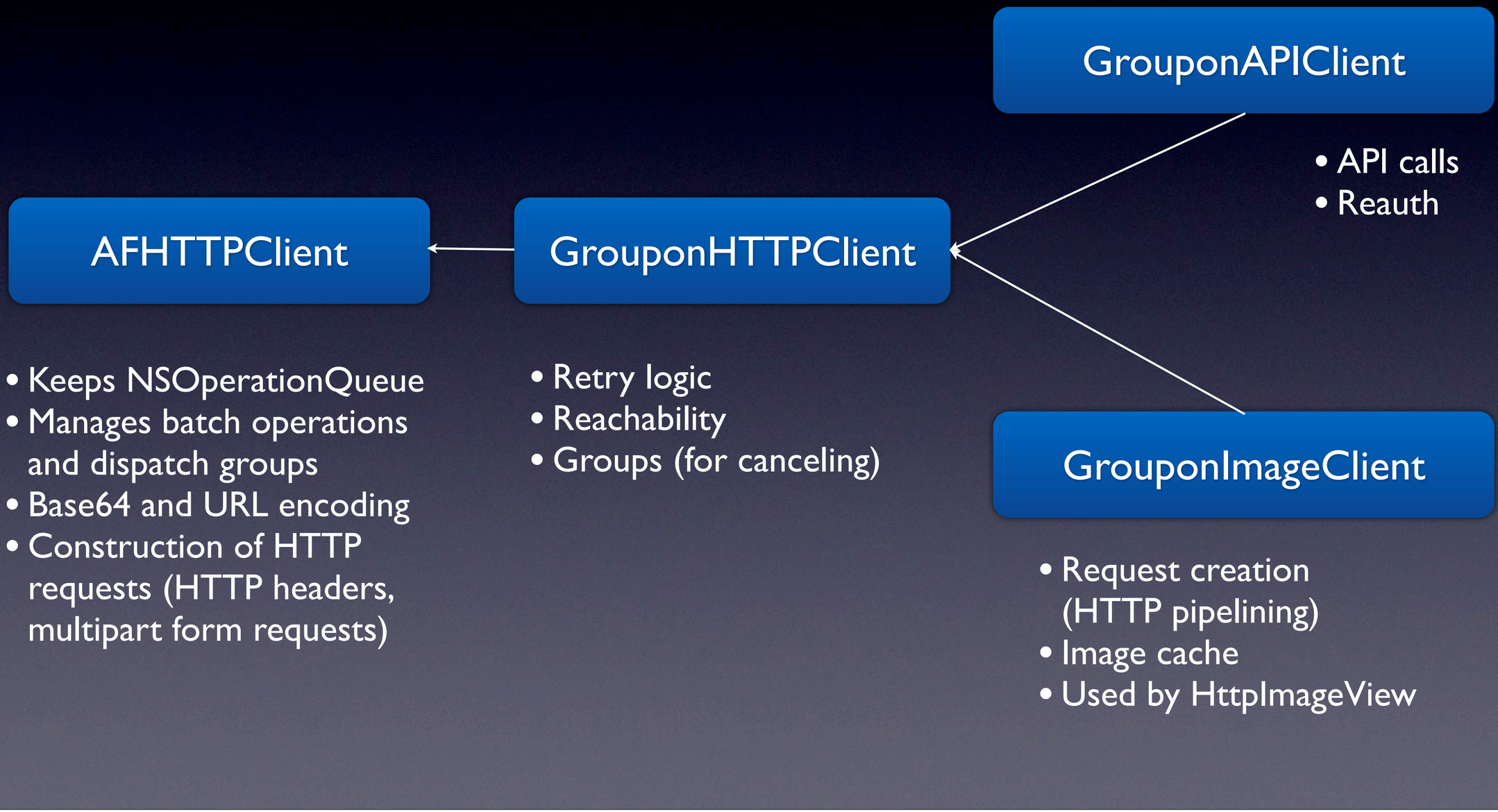
Source Code

- git checkout chapter_two - Images

UIImageView+AFNetworking



Clients


GrouponAPIClient
• API calls
• Reauth
AFHTTPClient
GrouponHTTPClient
• Keeps NSOperationQueue
• Manages batch operations and dispatch groups
• Base64 and URL encoding
• Construction of HTTP requests (HTTP headers, multipart form requests)
• Retry logic
• Reachability
• Groups (for canceling)
GrouponImageClient
• Request creation (HTTP pipelining)
• Image cache
• Used by HttpImageView

HttpImageView

- Spinner
- Loading
- URL
- Image request operation

Fast loading

- HTTP pipelining

```
NSMutableURLRequest *request = [NSMutableURLRequest requestWithURL:...];
[request setHTTPShouldUsePipelining:YES];
```

- 8 requests at a time (NSOperationQueue)

Old

New


QuickTime Player File Edit View Share Window Help
Sun 6:56 PM Andrey Yegorov
Screen Shot 2013-01....14.39 AM
Screen Shot 2013-02....32.33 PM
My Withings
AT&T 6:56 PM 23%
st For You
Getaways
Reward
4-Star Suites Overlooking Niagara Falls
C$321 C$119
680+ BOUGHT · NIAGARA
Luxury Hotel near US Capitol Building
$139 $97
320+ BOUGHT · WASHINGTON
All-Inclusive Beachfront Reso
AT&T 6:56 PM
Weddings
Getaways
Rewards
Glamorous 1950s Hotel in Sonoma Wine Country
$149 $104
10+ BOUGHT · SANTA ROSA
Luxury Hotel near US Capitol Building
$139 $97
320+ BOUGHT · WASHINGTON
Four-Star Hotel in New York's Times Square

AFImageRequestOperation

- Doesn't uncompress images to memory
- Scale images down on retina displays https://github.com/AFNetworking/AFNetworking/pull/53

Let's go over our TODO list

- Need a way to perform several operations at the same time
- Need a way to have a retry mechanism configurable per operation
- Need a good way to define dependencies between operations (CountryManager, channel creation etc)
- Need to have one point of entry for Groupon API transactions so we can regulate network load and prioritize operations
- Need a way to use blocks for simplicity

Need a way to perform several operations at the same time

- GrouponAPIClient is up to 4 operations at a time
- GrouponImageClient is up to 8 operations at a time
- Functionality is provided by NSOperationQueue and NSURLConnection already

Need a way to have a retry mechanism configurable per operation

```
GrouponHTTPRequestOperation *operation =
[[GrouponAPIClient sharedInstance] authorizeWithCredentials:inCredentials
            success:^(AFHTTPRequestOperation *operation, id responseObject) {
                ...
            } failure:^(AFHTTPRequestOperation *operation, NSError *error) {
                ...
            } completion:^{
                ...
            } group:@"login" startImmediately:NO];

[operation setRetryCountLimit:0]; // Do not try to re-authenticate

[[GrouponAPIClient sharedInstance] enqueueHTTPRequestOperation:operation];
```

Retry logic

- We need to make reusable copy of an operation to retry, since operation state had been altered
- But we cannot do it with completion blocks!

Retry logic

```
self.completionBlock = ^ {
    if (self.error) {
        if (failure) {
            dispatch_async(self.failureCallbackQueue ?: dispatch_get_main_queue(), ^{
                failure(self, self.error);
            });
        }
    } else {
        dispatch_async(json_request_operation_processing_queue(), ^{
            id JSON = self.responseJSON;
            if (self.JSONError) {
                if (failure) {
                    dispatch_async(self.failureCallbackQueue ?: dispatch_get_main_queue(),
^{
                        failure(self, self.error);
                    });
                }
            } else {
                if (success) {
                    dispatch_async(self.successCallbackQueue ?: dispatch_get_main_queue(),
^{
                        success(self, JSON);
                    });
                }
            }
        });
    }
};
```

Retry logic

```
GrouponHTTPRequestCompletionHandlerBlock completionHandler = ^(GrouponHTTPRequestOperation * instance) {

    if ([instance isCancelled]) {
        if (instance.dispatchGroup) {
            dispatch_group_leave(instance.dispatchGroup);
        }
        return;
    }

    if (instance.error) {
        if (failure) {
            dispatch_async(dispatch_get_main_queue(), ^(void) {
                failure(instance, instance.error);
                if (instance.dispatchGroup) {
                    dispatch_group_leave(instance.dispatchGroup);
                }
            });
        } else {
            if (instance.dispatchGroup) {
                dispatch_group_leave(instance.dispatchGroup);
            }
        }
    } else {
              ...
    }
};

__block id blockSelf = self;
self.completionBlock = ^{ completionHandler(blockSelf); };

self.backupOfCompletionHandler = completionHandler;
```

Retry logic

```
- (id)copyWithZone:(NSZone *)zone {
    GrouponHTTPRequestOperation *operation = [[[self class] alloc]
initWithRequest:self.request];

    operation.completionBlock = ^{ self.backupOfCompletionHandler(operation); };
    operation.backupOfCompletionHandler = self.backupOfCompletionHandler;
    [operation setQueuePriority:NSOperationQueuePriorityVeryHigh];
    [operation setRetryCount:[self retryCount]];
    [operation setRetryCountLimit:[self retryCountLimit]];
    [operation setRetryCondition:[self retryCondition]];
    [operation setGroupId:self.groupId];

    if (self.dispatchGroup) {
        operation.dispatchGroup = self.dispatchGroup;
        dispatch_group_enter(operation.dispatchGroup);
    }

    return operation;
}
```

Retry logic

- It looks like we have been right about retrying:

https://github.com/AFNetworking/AFNetworking/issues/596

Need a good way to define dependencies between operations

```
NSMutableArray *countryOperations = [NSMutableArray array];

// 1 - status operation
GrouponHTTPRequestOperation *statusOperation =
[[GrouponAPIClient sharedInstance] apiStatusWithLocation:... group:SET_COUNTRY_OPS_GROUP startImmediately:NO];
[countryOperations addObject:statusOperation];

// 2 - country info operation
GrouponHTTPRequestOperation *countryInfoOperation =
[[GrouponAPIClient sharedInstance] countryInfoWithPriority:... group:SET_COUNTRY_OPS_GROUP
startImmediately:NO];
[countryOperations addObject:countryInfoOperation];

...

// 4 - divisions operation
GrouponHTTPRequestOperation *divisionsOperation =
[[GrouponAPIClient sharedInstance] divisionsWithPriority:... group:SET_COUNTRY_OPS_GROUP startImmediately:NO];
[countryOperations addObject:divisionsOperation];

[[GrouponAPIClient sharedInstance] enqueueBatchOfHTTPRequestOperations:countryOperations ...
completionBlock:^(NSArray *operations) {
    if (inBlock) {
        inBlock(YES);
    }
     [[NSNotificationCenter defaultCenter] postNotificationName:CountryChangeEndedNotification object:nil
userInfo:nil];
}];
```

Need a way to use blocks for simplicity

```
- (void)downloadAndComposeImagesForPromos:(BOOL)isBottom {
      NSArray *promos = (isBottom) ? m_bottomPromos : m_middlePromos;
      BOOL isFullWidth = ([promos count] == 1);
      for (PromoObj* promo in promos) {
            ...
            UIImage *img = [_app.imageCache getImage:url defaultImage:nil priority:DOWNLOADER_PRIORITY_TOP_MOVEABLE];
            if (!img && url) { // make sure we have url when adding promo data
                  [m_requestedPromoImages setValue:[NSDictionary dictionaryWithObjectsAndKeys:
                                                                  [NSNumber numberWithBool:isFullWidth], @"fullWitdh",
                                                                  [NSNumber numberWithBool:isBottom], @"bottom",
                                                                  promo, @"promo",
                                                                  nil]
                                                          forKey:url];
      }
      ...
}

#pragma mark - ImageCacheDelegate methods
- (void)imageReady:(UIImage*)image urlString:(NSString*)urlStr statusCode:(int)statusCode {
      NSDictionary *dict = [m_requestedPromoImages dictionaryForKey:urlStr defaultValue:nil];
      if (dict) {
            PromoObj *promo = [dict valueForKey:@"promo"];

            if ([dict objectForKey:@"header"]) {
                  if ( (image) && (promo) )
                        [self composeAndSavePromoHeaderImagesFromAPIImage:image promo:promo];
            } else {
                  BOOL isFullSize = [dict boolForKey:@"fullWitdh" defaultValue:YES];
                  BOOL isBottom = [dict boolForKey:@"bottom" defaultValue:YES];
                  PromoObj *promo = [dict valueForKey:@"promo"];
                  if ( (image) && (promo) ) {
                        [self composeAndSavePromoButtonImagesFromAPIImage:image fullSize:isFullSize bottom:isBottom
promo:promo shouldPostNotification:YES];
                  }
            }
            [m_requestedPromoImages removeObjectForKey:urlStr];
      }
}
```

Need a way to use blocks for simplicity

```
- (void)downloadAndComposeImagesForPromos:(BOOL)isBottom {
    NSArray *promos = (isBottom) ? m_bottomPromos : m_middlePromos;
    BOOL isFullWidth = ([promos count] == 1);
    for (PromoObj* promo in promos) {

          ...

        [[GrouponImageClient sharedInstance] imageForURL:[NSURL URLWithString:url]
                                                 success:^(..., UIImage *image) {

                [self composeAndSavePromoButtonImagesFromAPIImage:image
                fullSize:isFullWidth
                bottom:isBottom
                promo:promo];
                                                } failure:nil];
    }
    [[GrouponImageClient sharedInstance] imageForURL:[NSURL URLWithString:url]
                success:^(..., UIImage *image) {
                [self composeAndSavePromoHeaderImagesFromAPIImage:image promo:promo];
                } failure:nil];
}
```

Problems with AFNetworking

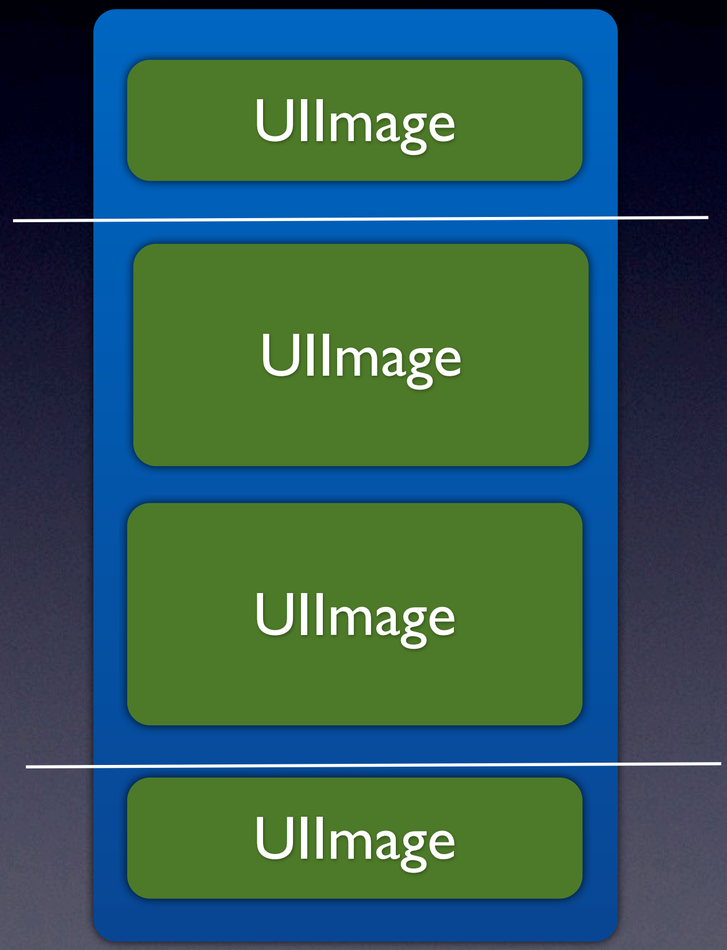
AFNetworking Problems

- "Dealloc Problem" (still not fixed?) https://github.com/AFNetworking/AFNetworking/issues/56
- Dispatch groups (fixed in our operations) https://github.com/AFNetworking/AFNetworking/issues/362
- Retry, reauth and cancel logic is complicated (fixed, but still complicated - fragile to changes)

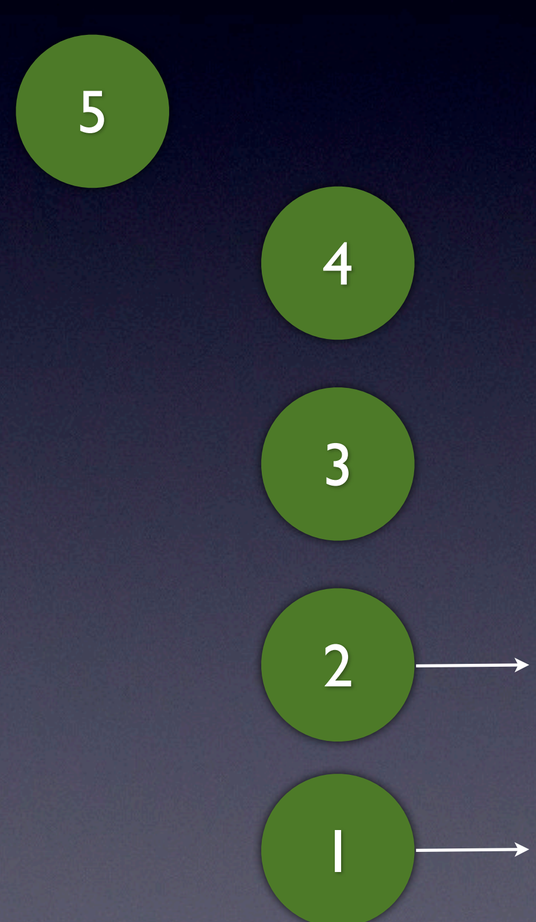
AFNetworking Problems

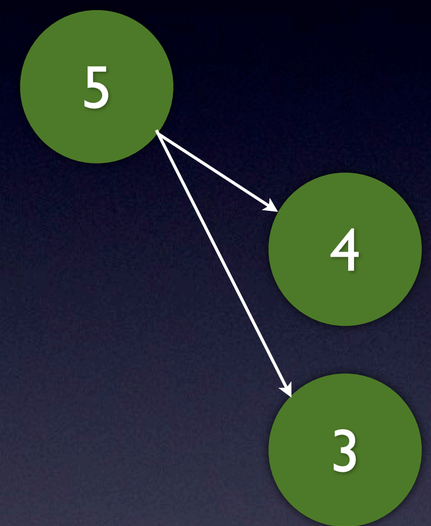
- Locks: https://github.com/AFNetworking/AFNetworking/pull/131
- Getting fancy with NSOperationQueue - NSOperationStack https://github.com/AFNetworking/AFNetworking/issues/798

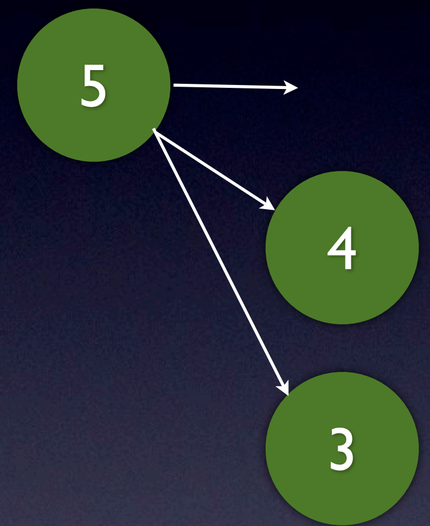
NSOperationStack



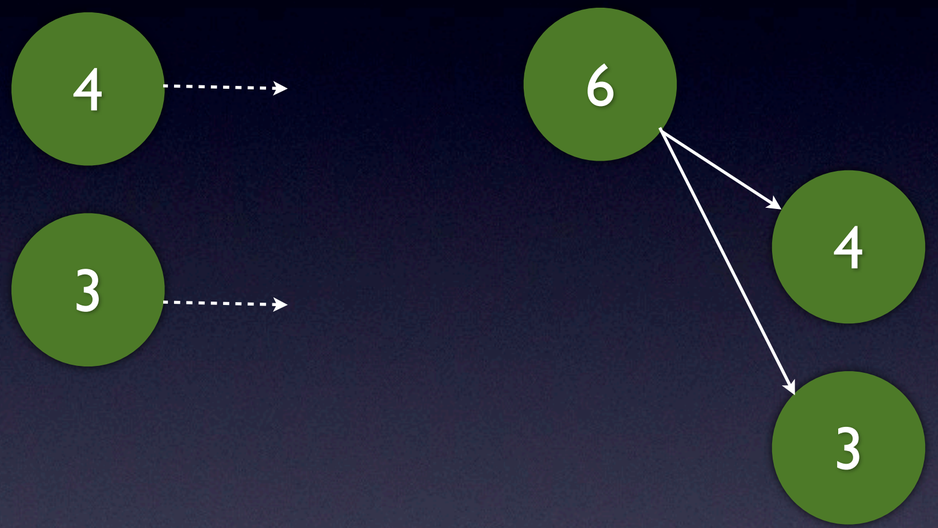
NSOperationStack







NSOperationStack

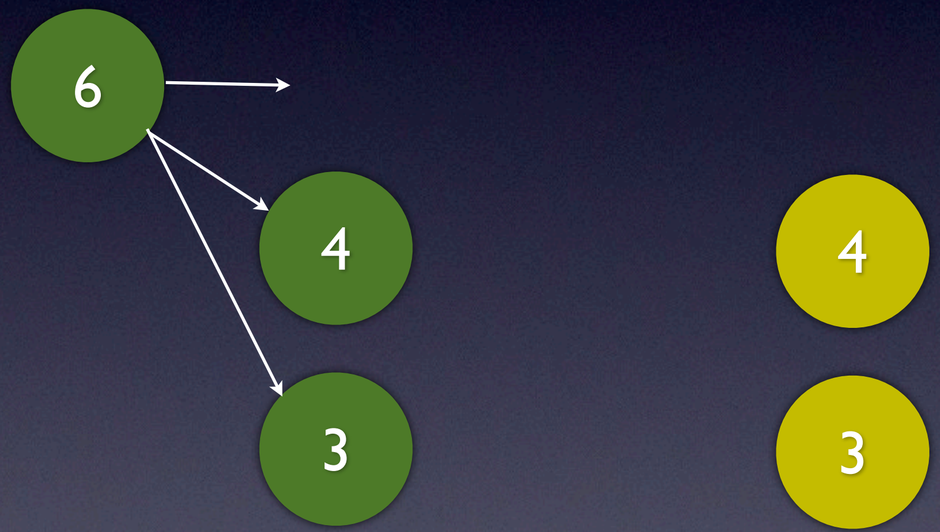


NSOperationQueue

NSOperation::start

NSOperation::isReady

NSOperationStack



Don’t change dependencies for operations which are already in the queue!

How to test your networking code

OMG!

They got me on 401 again

Source Code

- git checkout chapter_three - Testing

Testing

- Kiwi: https://github.com/allending/Kiwi
- Nocilla: https://github.com/luisobo/Nocilla

Questions?