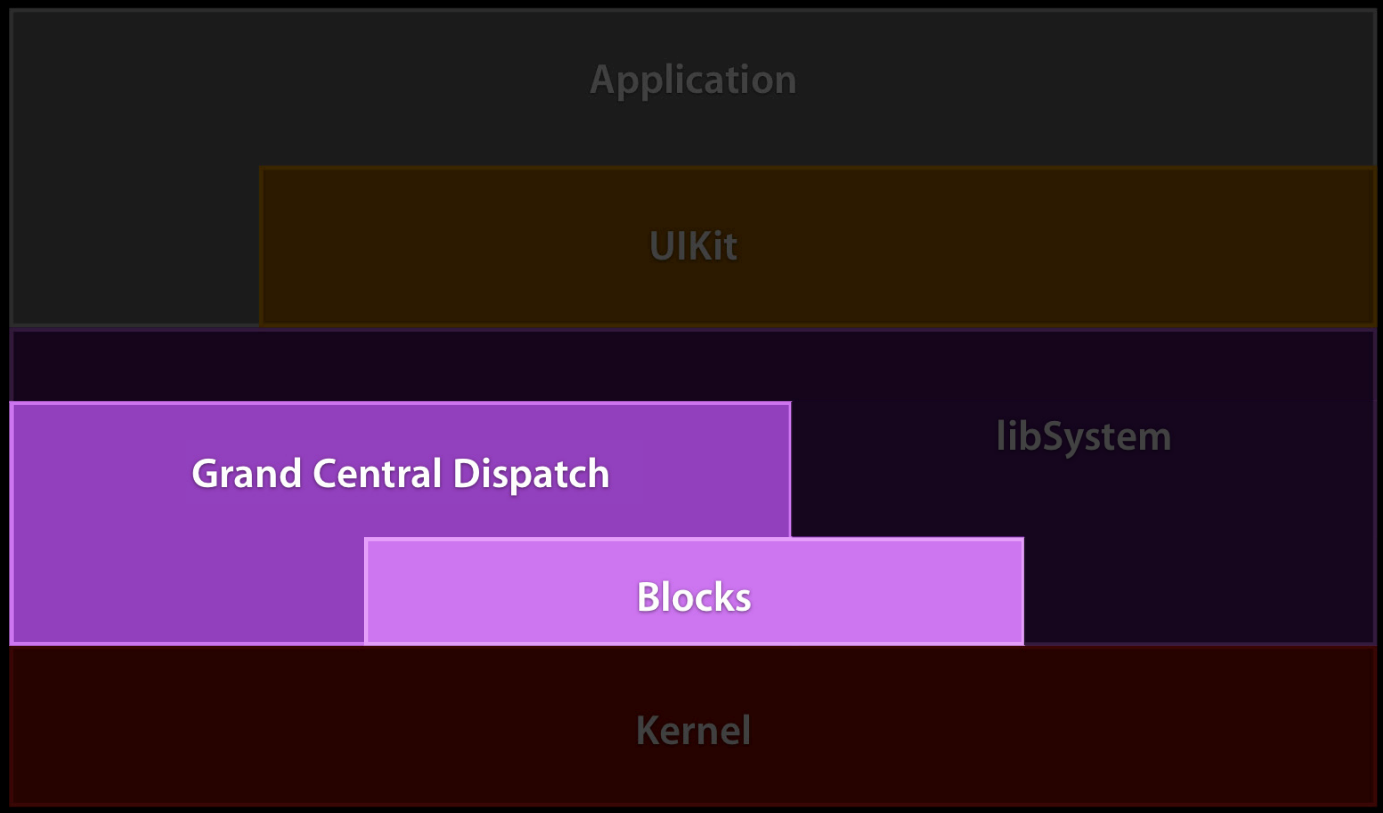
Introducing Blocks and Grand Central Dispatch on iPhone

Kevin van Vechten
Core OS

Technology Stack



Blocks

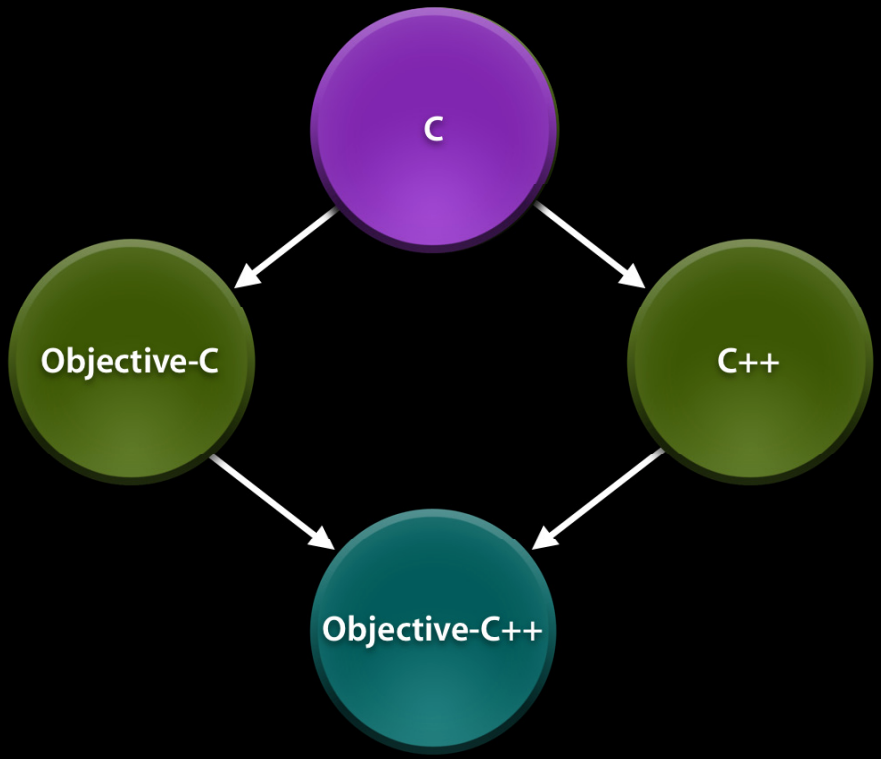
Bill Bumgarner

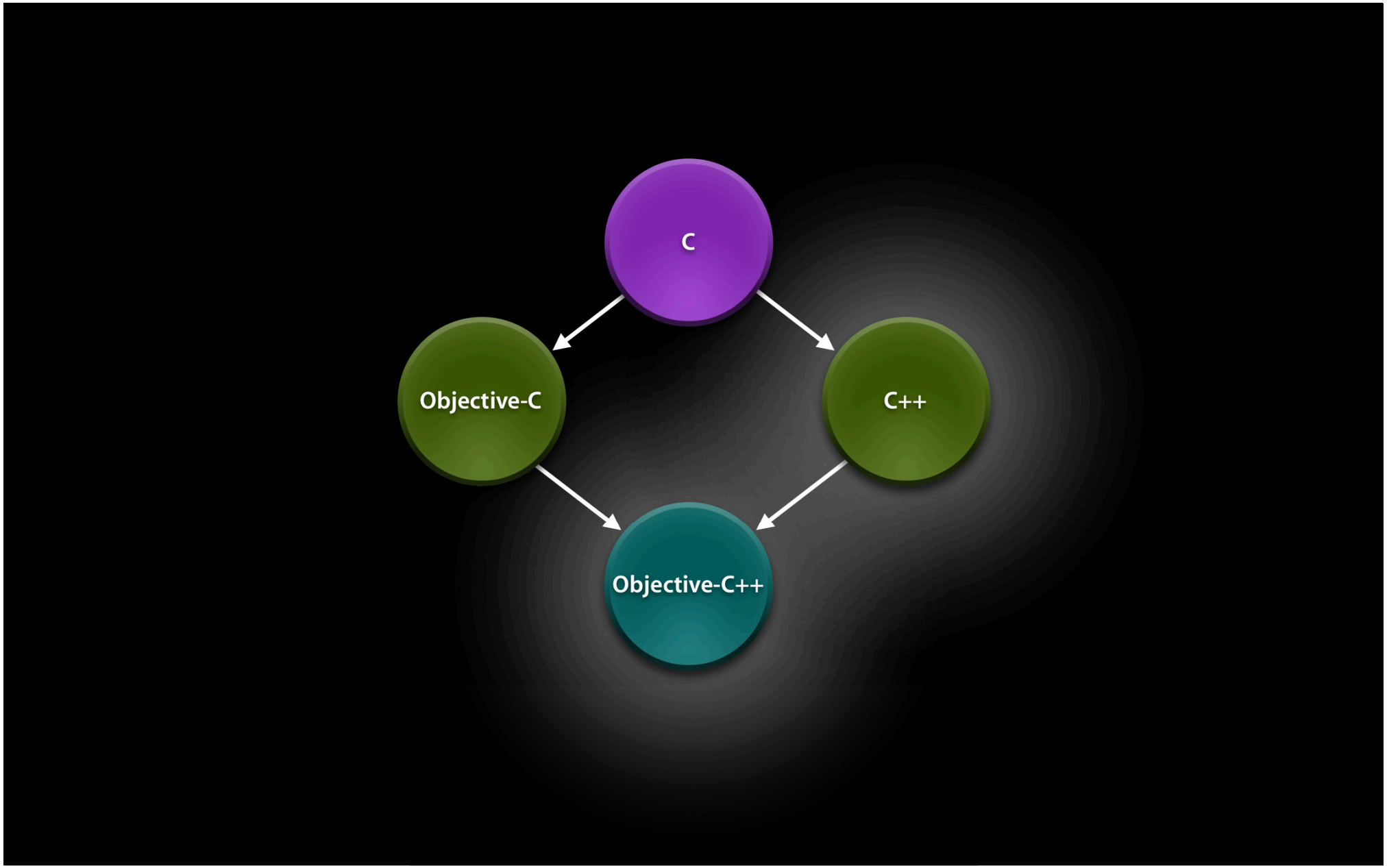
```
(lambda (a) (add a d))
```


```
10 timesRepeat:[pen turn:d; draw]
```


```
z.each {|val| puts(val + d.to_s)}
```


```
repeat(10, ^{ putc('0'+ d); });
```


C
Objective-C
C++
Objective-C++


C
Objective-C
C++
Objective-C++

Basic Blocks

Block Literals

```
[mySet objectsPassingTest:^(id obj, BOOL *stop) {
  return [obj testValue: value];
}];


dispatch_async(main_queue, ^{
            [viewController displayNewStuff];
});
```

Block Literal Syntax


```
^ void (id item) { [item doOneGoodThing]; }

    ^ (id item) { [item doOneGoodThing]; }

^ void (void) { [local doTwoGoodThings]; }

          ^ { [local doTwoGoodThings]; }
```

Blocks as Data

Function Pointers

```
void (*callable)(void);
```

Block Pointers

```
void (^callable)(void);
```

Ugly Block Pointers

```
char *(^worker)(char *a,
                BOOL(^done)(int));
```

Typedefs Are Your Friend

```
typedef BOOL (^doneBlk_t)(int);

char *(^workB)(char *a,
               doneBlk_t d);
```

Blocks in Practice

Worker Block

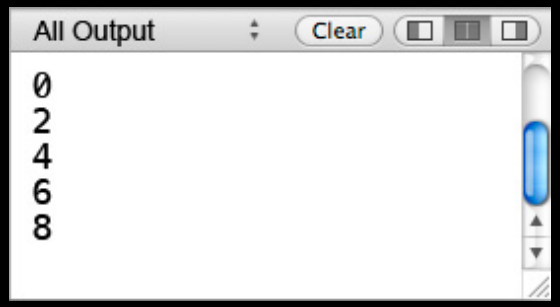
```
typedef void (^workBlk_t)(int i);
```

Work Block Consumer

```
typedef void (^workBlk_t)(int i);
void
repeat(int n, workBlk_t aBlock) {
     for (int i = 0; i < n; ++i)
         aBlock(i);
}
```

Putting It Together

```
int d = 2;
workBlk_t w = ^(int i){
    printf(“%d\n", i * d);
};
repeat(5, w);
```



Output

iOS 4

Over 100 APIs

Multitasking

Foundation

Grand Central Dispatch

Four Common Patterns

Synchronous Execution

```
[mySet objectsPassingTest: ^(id obj, BOOL *stop) {
      return [obj testValue: value];
}];

[aDictionary enumerateKeysAndObjectsUsingBlock: ^(id k, id v, BOOL *stop) {
      NSLog(@"%@ => %@", k, v);
}];
```

Callbacks

```
[application beginBackgroundTaskWithExpirationHandler: ^{
      /* expiration handler callback code */
}];

[anOperation setCompletionBlock: ^{
      /* handle operation completion */
}];
```

Asynchronous Execution

```
[operationQueue addOperationWithBlock: ^{
      /* hard work is hard */
}];

dispatch_async(main_queue, ^{
      [viewController displayNewStuff];
});
```

Lockless Exclusion

```
// thread a
dispatch_async(queue, ^{ ... });

// thread b
dispatch_sync(queue, ^{ ... });

// main thread
dispatch_async(queue, ^{ ... });
```

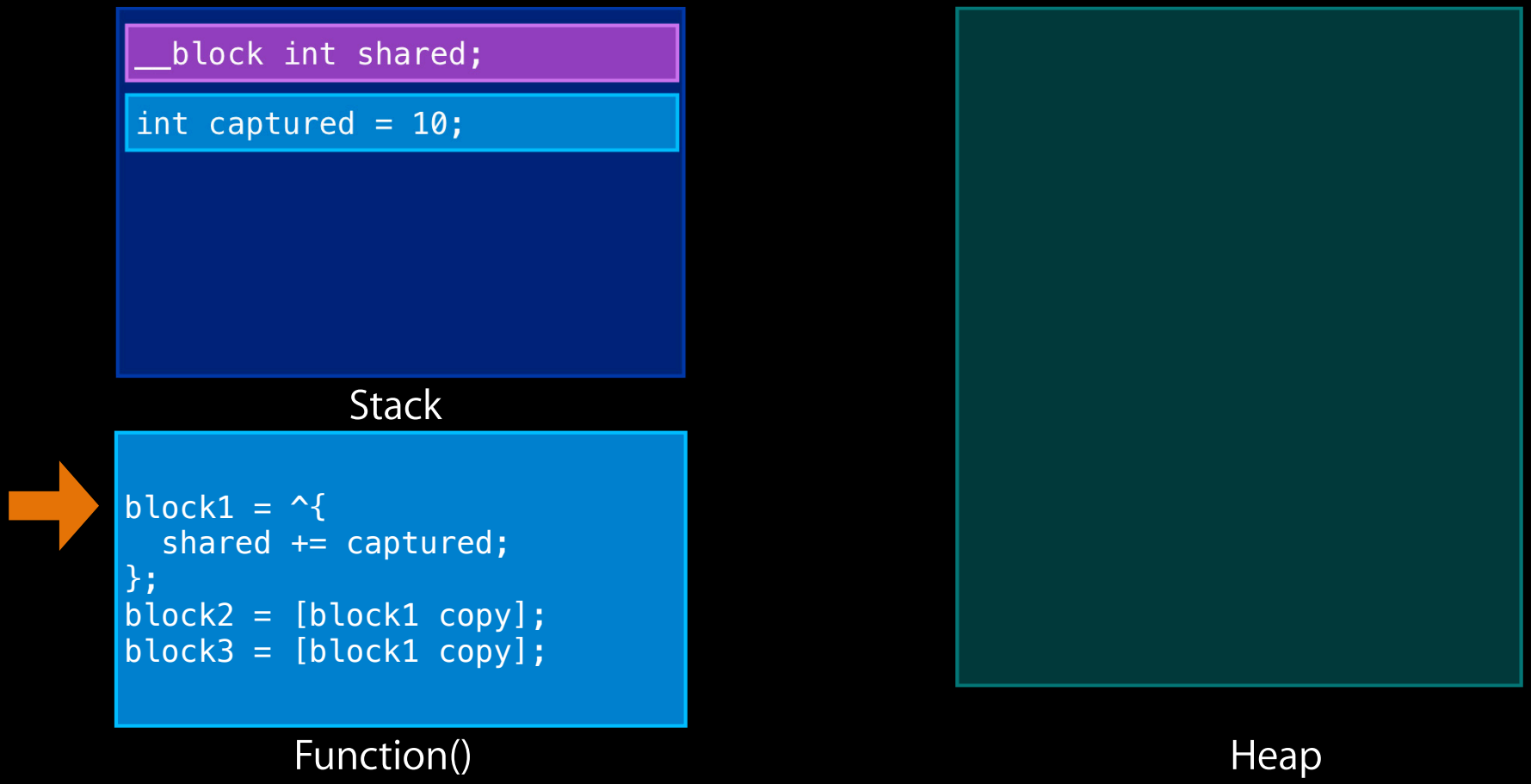
Blocks in Detail

Block Object Details

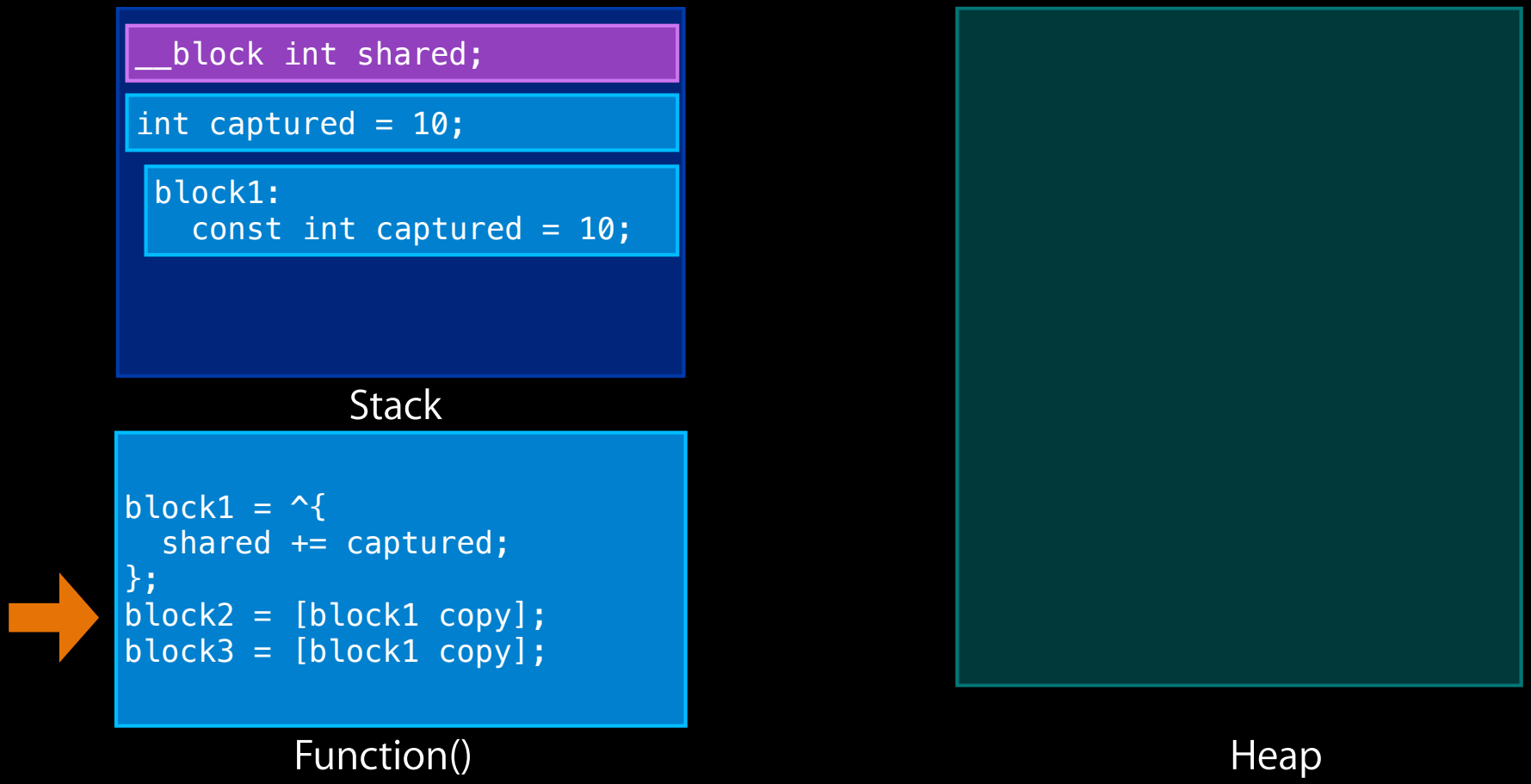
- Blocks are Objective-C objects
- Block objects start out on the stack
  - Copied to the heap using `[aBlock copy]` or `Block_copy()`
  - Release with `[aBlock release]` or `Block_release()`
- Blocks have private const copy of stack (auto) variables
  - Object references are retained
- Mutable variables must be declared with `__block` keyword
  - Shared with all other blocks that use that variable
  - Shared with the scope of declaration
  - `__block` Object references are not retained

Block Lifetime Illustrated

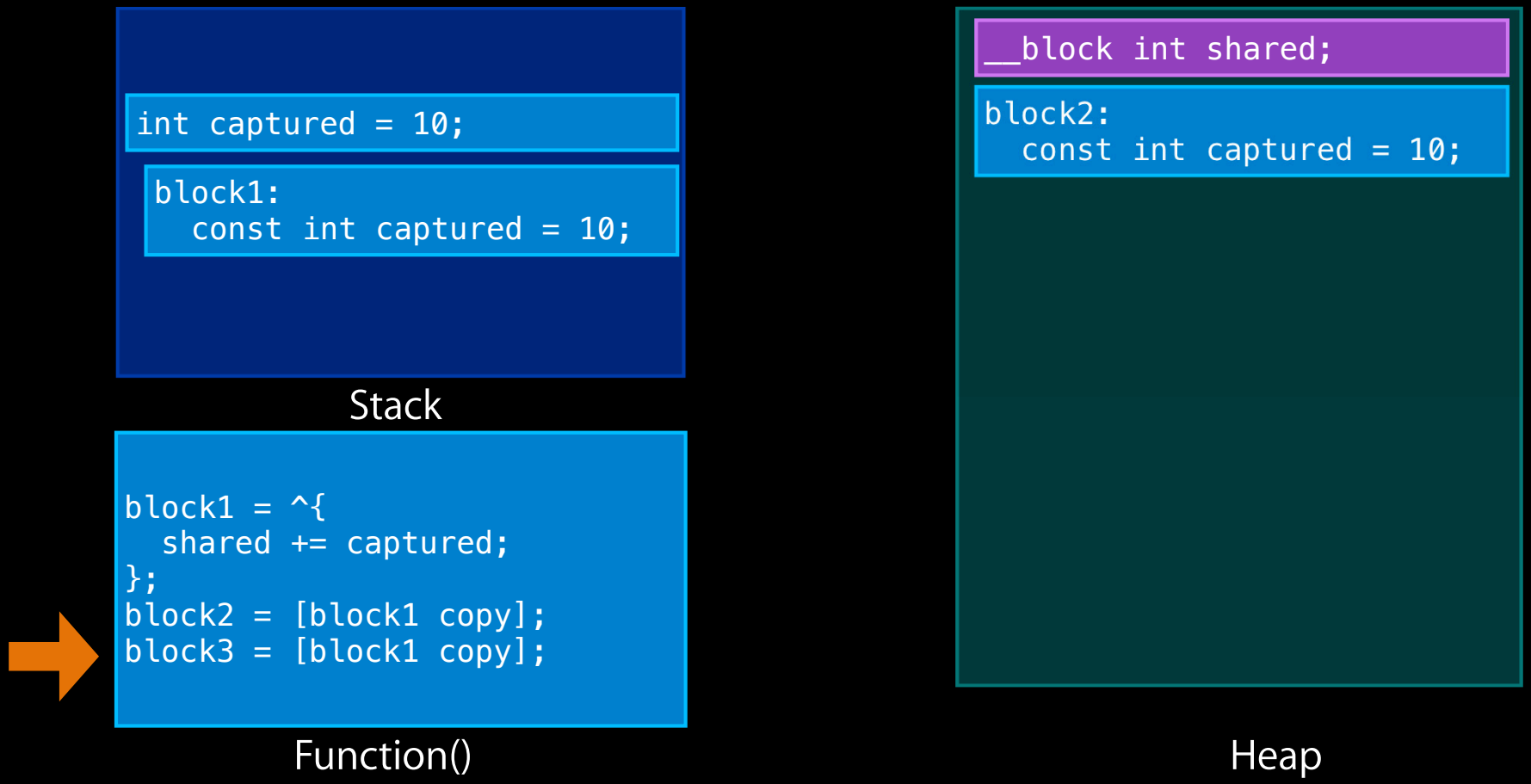
Start on Stack



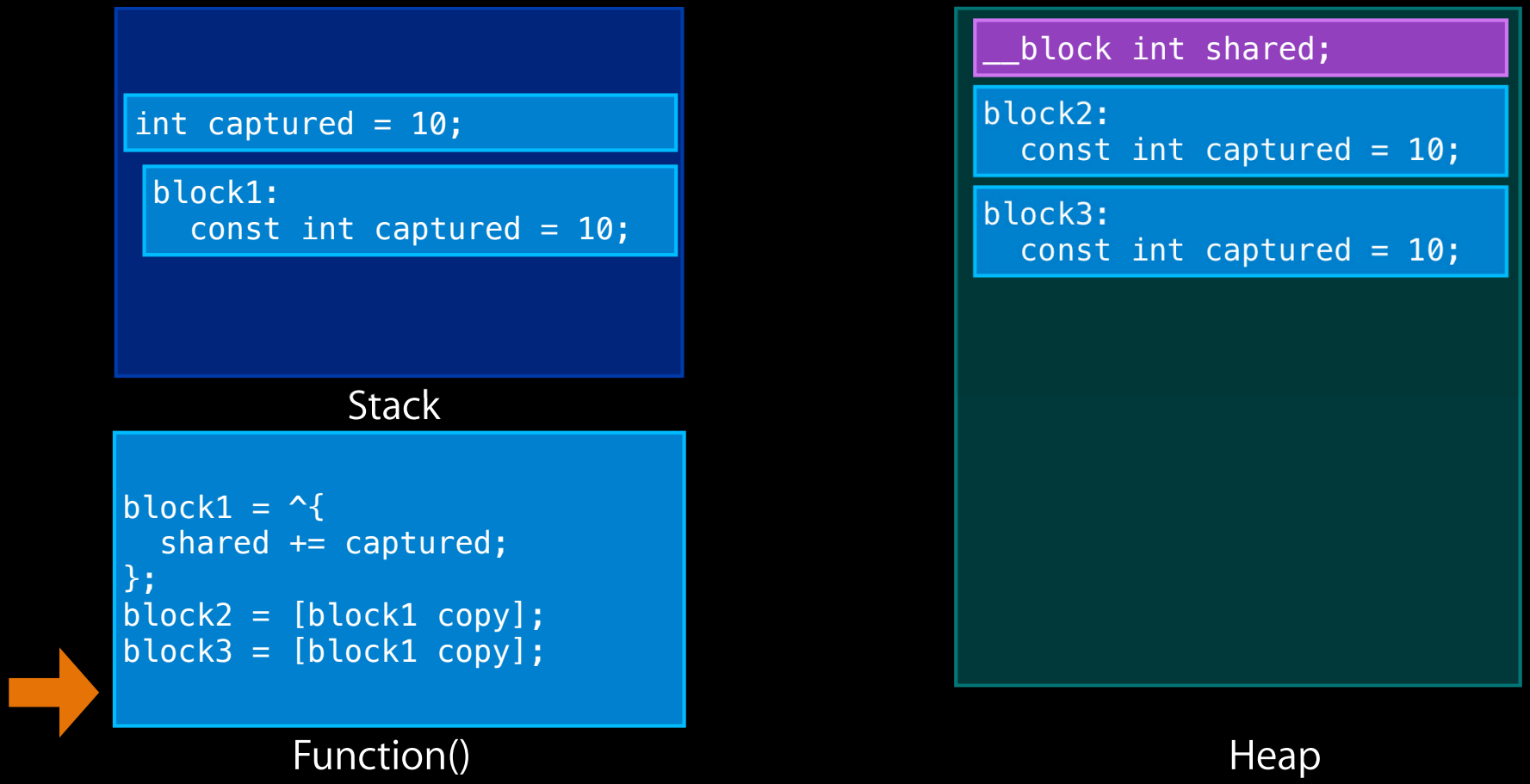
Capture State



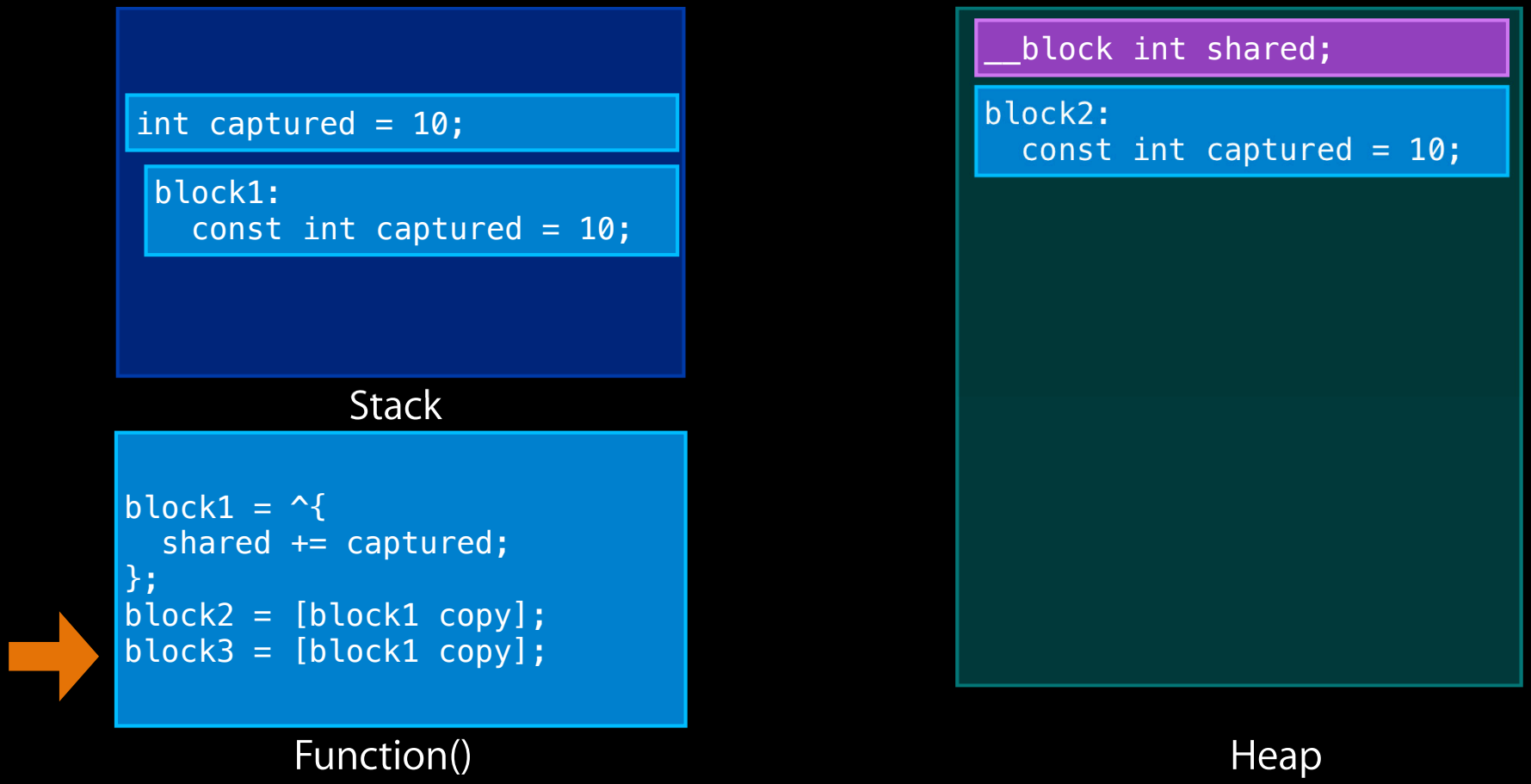
Copies State to Heap



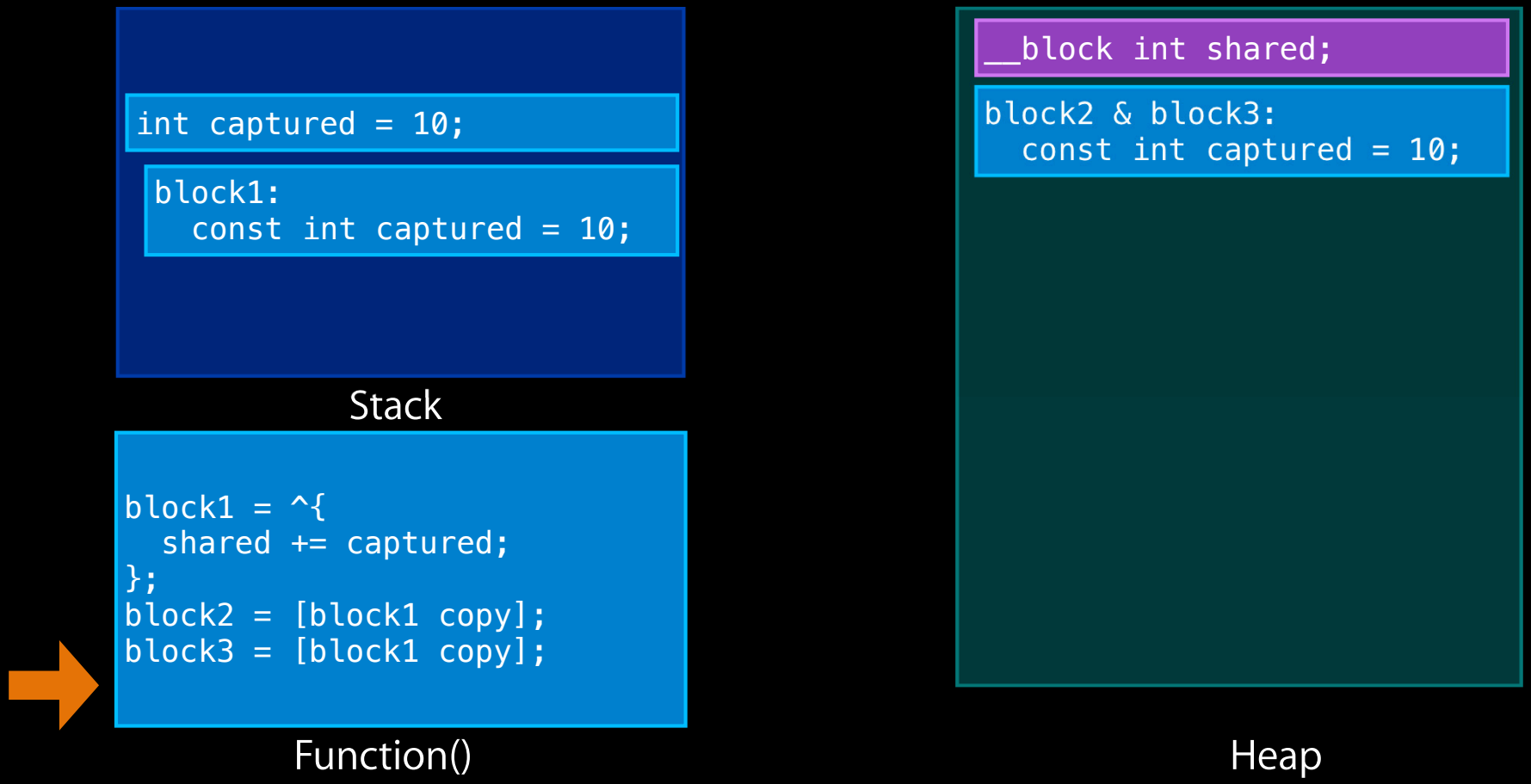
⚠May Produce Multiple Copies⚠



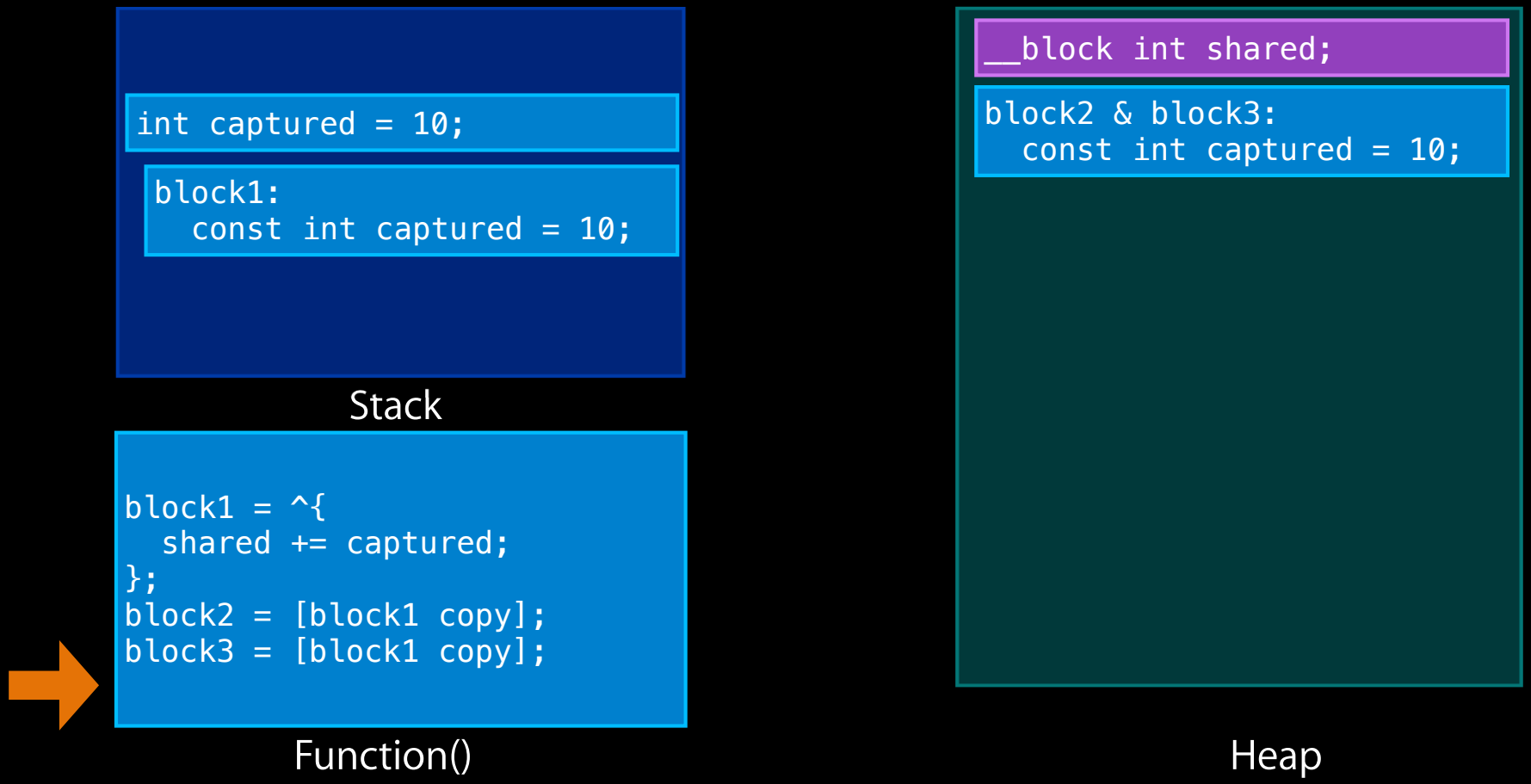
Avoid Copies



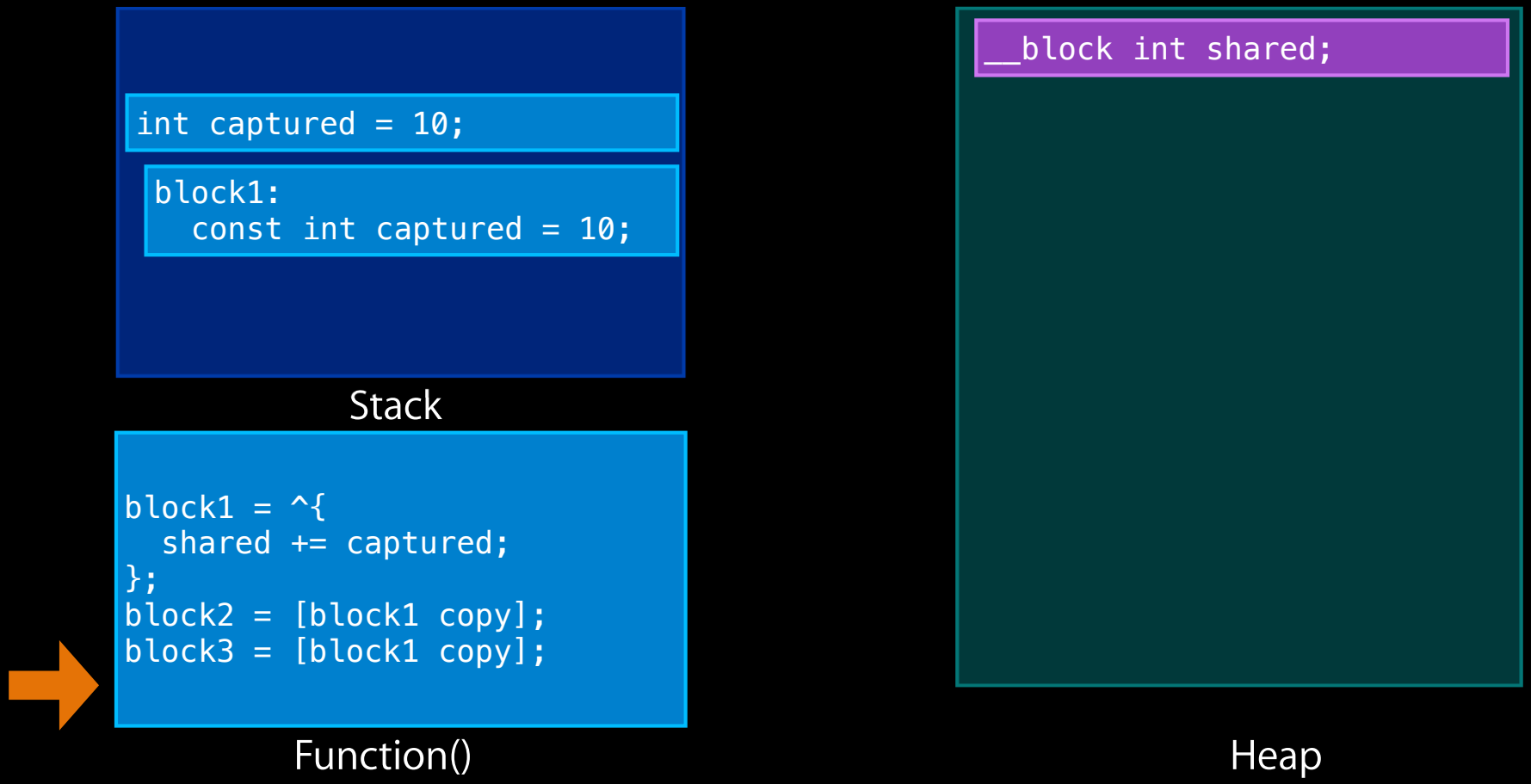
Bump Retain Count



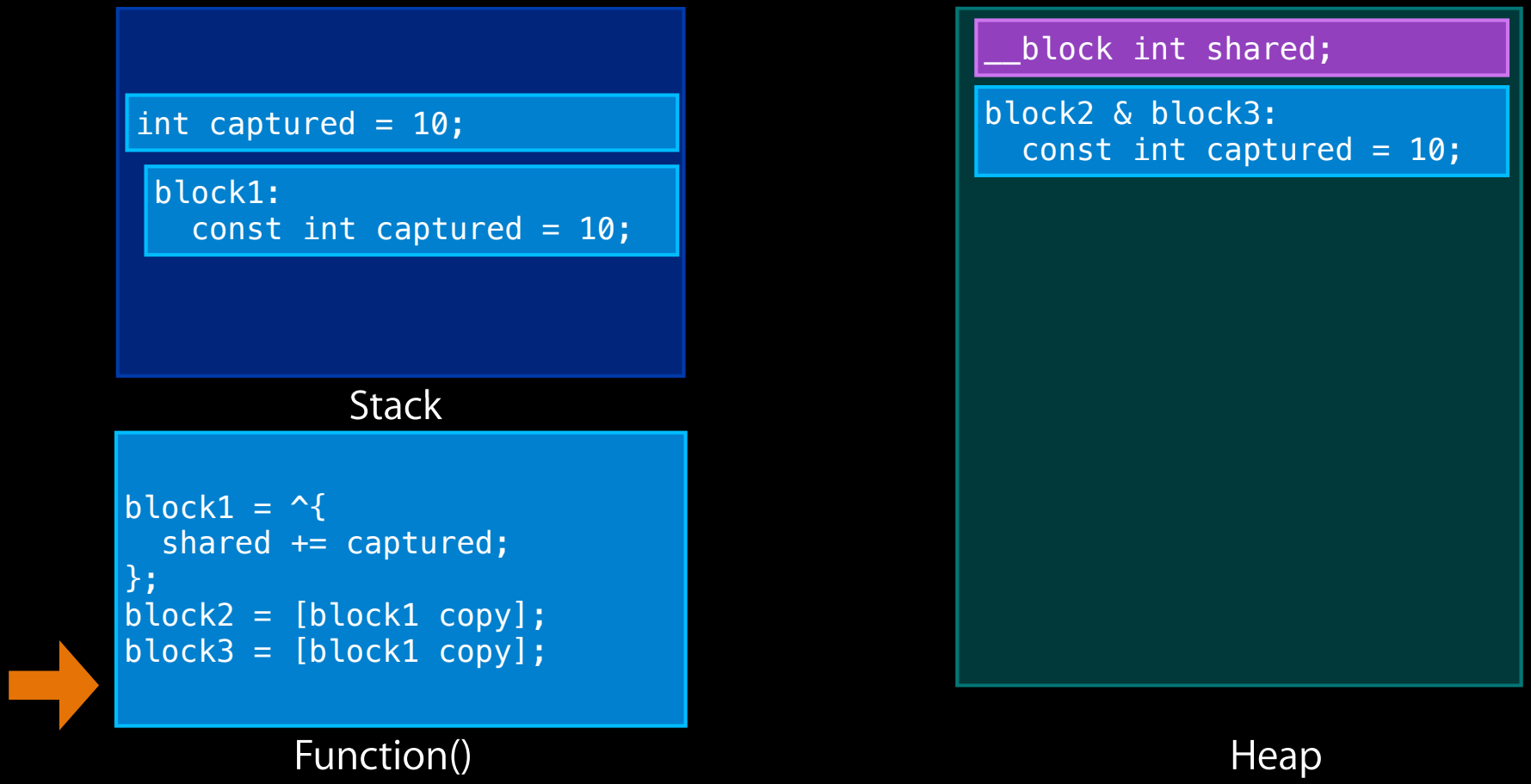
block2 / block3 Finish First



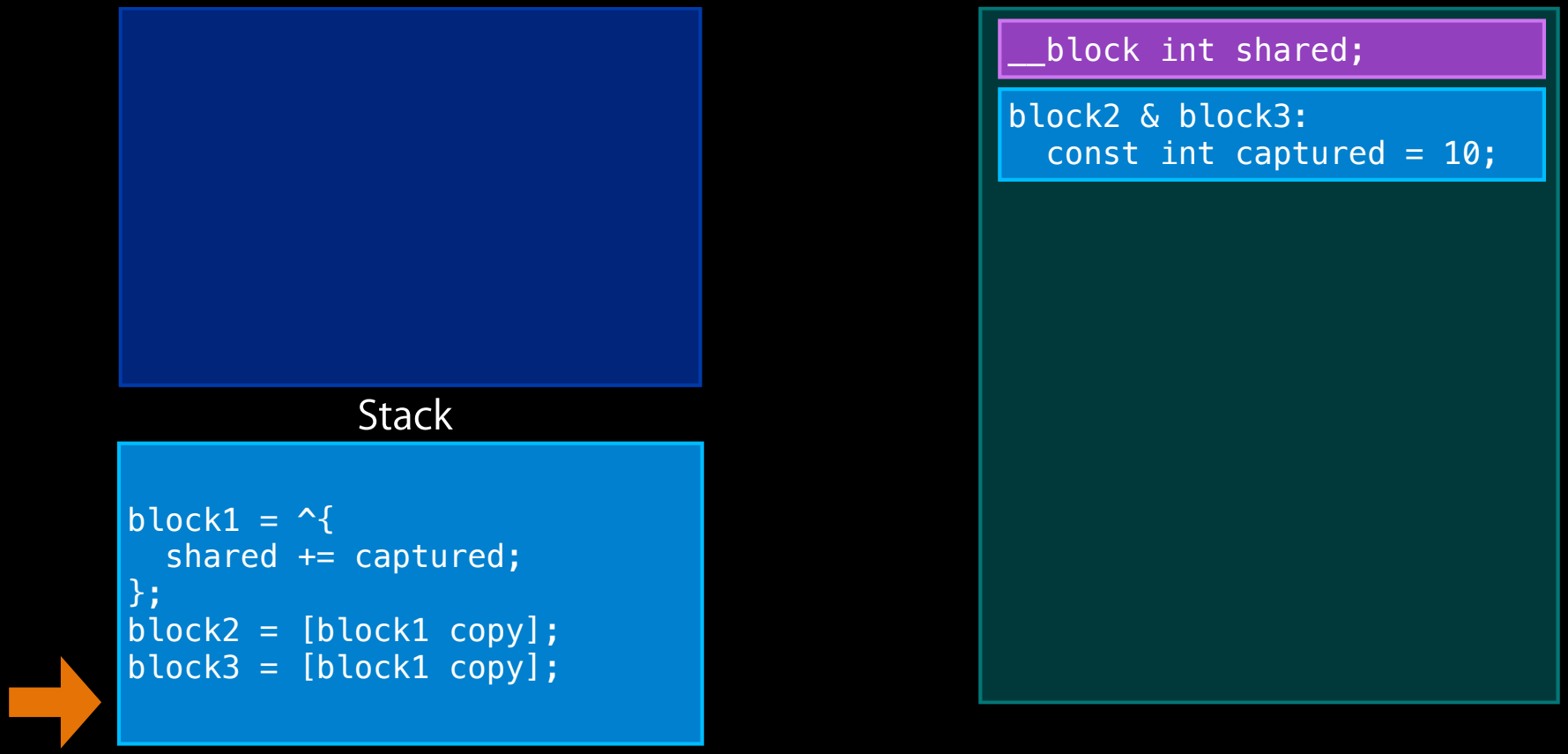
block2 / block3 Finish First



Function()/block1 Finishes First



Function()/block1 Finishes First



Grand Central Dispatch

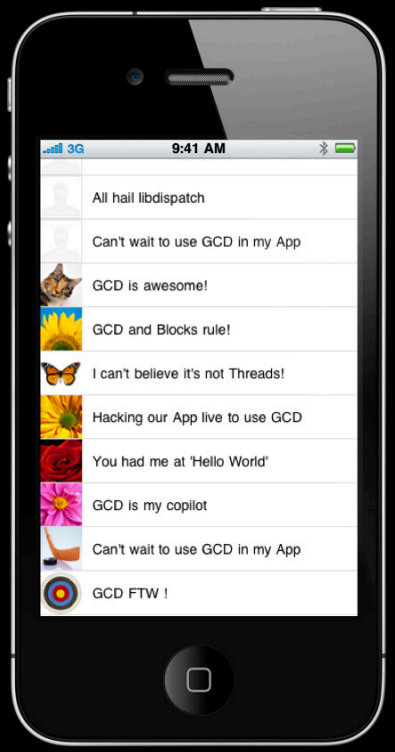
Shiva Bhattacharjee
Core OS

Grand Central Dispatch

- Threading is hard
- Using GCD makes it simple and fun
- No explicit thread management
- Extremely efficient under the hood
- GCD and Blocks are a powerful duo

Grand Central Dispatch

Keeping your app responsive



Demo

Shiva Bhattacharjee

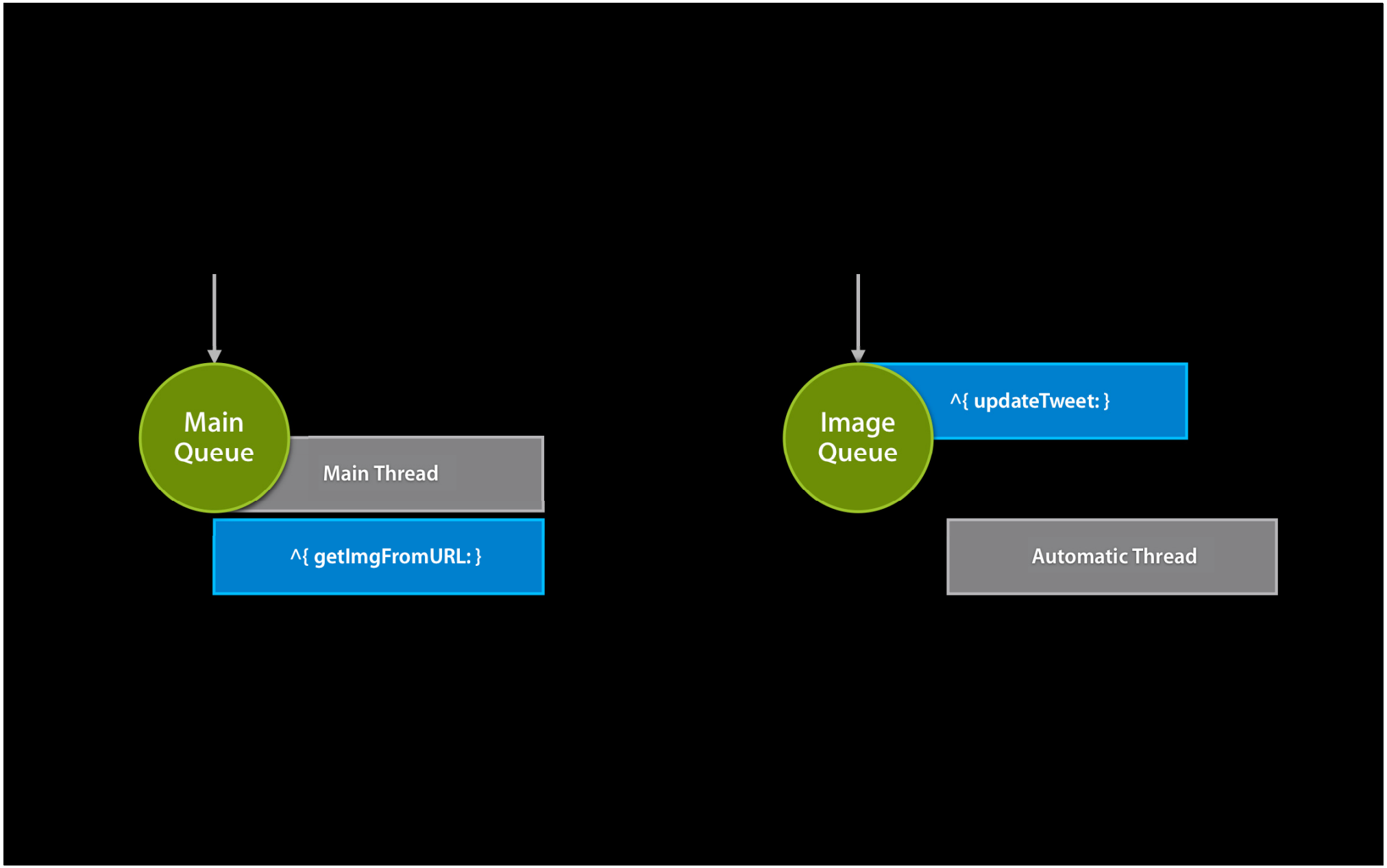
Keeping Your App Responsive

- Do not block the main thread
- Move work to another thread
- Update UI back on main thread

GCD

Keeping Your App Responsive

```
- (void)addTweetWithMsg:(NSString*)msg url:(NSURL*)url {
    // Controller UI callback on main thread
    DTweet *tw = [[DTweet alloc] initWithMsg:msg];
    [tweets addTweet:tw display:YES];
    dispatch_async(image_queue, ^{
    tw.img = [imageCache getImgFromURL:url];
        dispatch_async(main_queue, ^{
    [tweets updateTweet:tw display:YES];
        });
    });
    [tw release];
}
```


Main Queue
Main Thread
^{ getImgFromURL: }
Image Queue
^{ updateTweet: }
Automatic Thread

Keeping Your App Responsive

- Do not block the main thread
- Move work to another thread
- Update UI back on main thread

dispatch_async()

GCD Queues

Daniel Steffen
Core OS

GCD Queues

- Lightweight list of blocks
- Enqueue/dequeue is FIFO
- Enqueue with dispatch_async()
- Dequeue by automatic thread or main thread

Main Queue

Main Queue

- Executes blocks one at a time on main thread
- Cooperates with the UIKit main run loop
- dispatch_get_main_queue()

Main Queue

```
- (void)addTweetWithMsg:(NSString*)msg url:(NSURL*)url {
    // Controller UI callback on main thread
    DTweet *tw = [[DTweet alloc] initWithMsg:msg];
    [tweets addTweet:tw display:YES];
    tw.img = [imageCache getImgFromURL:url];
    [tweets updateTweet:tw display:YES];
    [tw release];
}
```

Main Queue

```
- (void)showTweetWithMsg:(NSString*)msg url:(NSURL*)url {
    // Controller networkIO callback on background thread
    id d = [NSDictionary dictionaryWithObjectsAndKeys:
            msg, @"msg", url, @"url", nil];
    [self performSelectorOnMainThread:
            @selector(updateTWDisplay:) withObject:d
            waitUntilDone:NO];
}
- (void)updateTWDisplay:(NSDictionary*)d {
    [self addTweetWithMsg:[d objectForKey:@"msg"]
            url:[d objectForKey:@"url"]];
}
```

Main Queue

```
- (void)showTweetWithMsg:(NSString*)msg url:(NSURL*)url {
    // Controller networkIO callback on background thread
    dispatch_async(dispatch_get_main_queue(), ^{
        id d = [NSDictionary dictionaryWithObjectsAndKeys:
                msg, @"msg", url, @"url", nil];
        [self performSelectorOnMainThread:
                @selector(updateTWDisplay:) withObject:d
                waitUntilDone:NO];
    });
}
- (void)updateTWDisplay:(NSDictionary*)d {
    [self addTweetWithMsg:[d objectForKey:@"msg"]
                url:[d objectForKey:@"url"]];
}
```

Creating Your Own Queues

Creating Your Own Queues

- Execute blocks one at a time
- On automatic helper thread
- “Queue up” background work

Creating Your Own Queues

```
dispatch_queue_t queue;

queue = dispatch_queue_create("com.example.purpose", NULL);

dispatch_release(queue);
```

Creating Your Own Queues

But wait, there's even more…

Queues Instead of Locks

- Enqueuing is thread-safe
- Execution is serial
- Protect access to shared data
- Queues are lightweight

Queues Instead of Locks

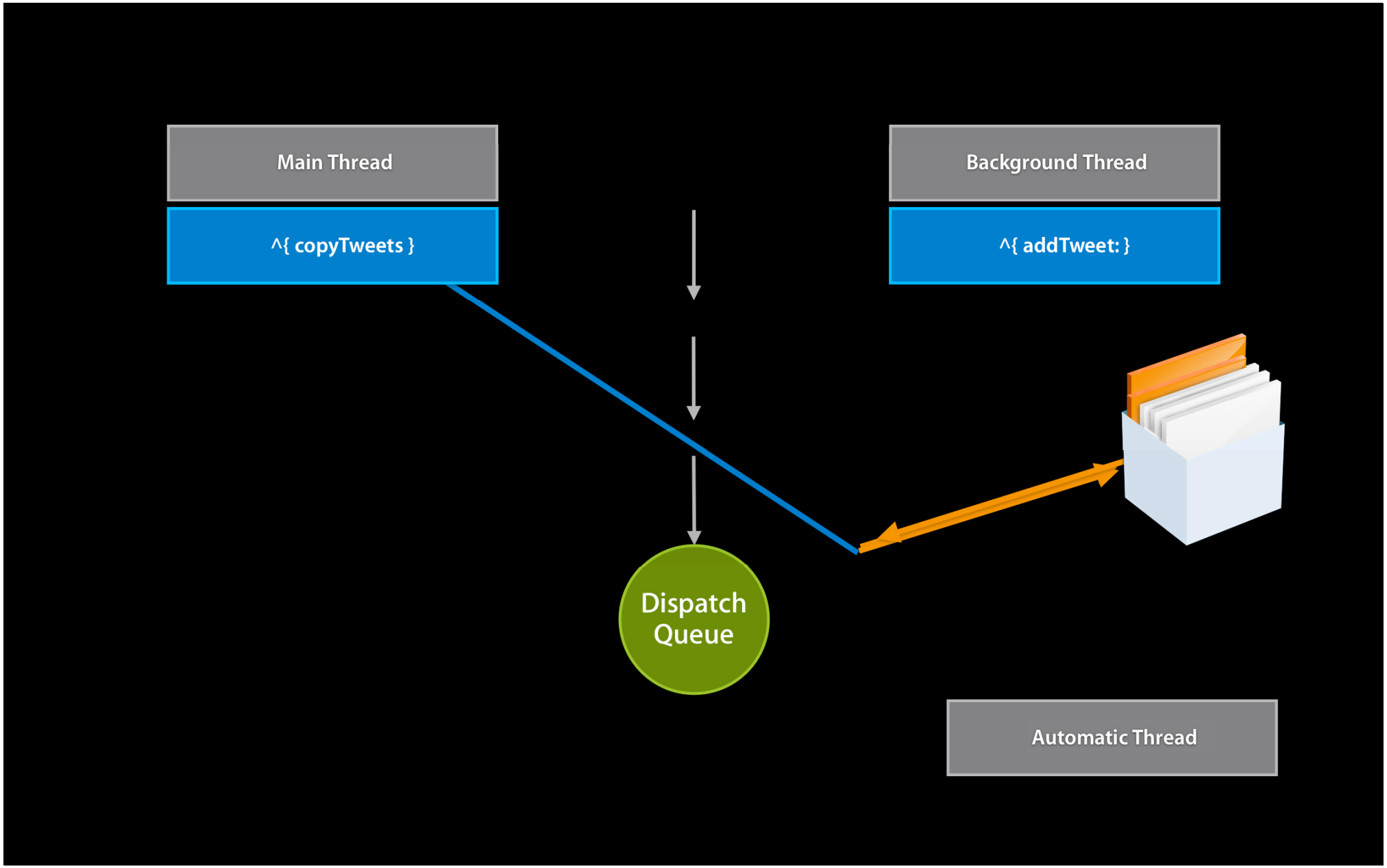
```
queue = dispatch_queue_create(“com.example.tweets”, NULL);
```

```
// Main Thread
dispatch_async(queue, ^{
    [tweets removeLastTweet];
});
```

```
__block NSArray *a;
dispatch_sync(queue, ^{
    a = [tweets copyTweets];
});
```

```
// Background Thread
dispatch_async(queue, ^{
    [tweets addTweet:tw];
});
```

```
dispatch_release(queue);
```


Main Thread
^{ copyTweets }
Background Thread
^{ addTweet: }
Dispatch Queue
Automatic Thread

Managing Queue Lifetime

Managing Queue Lifetime

- Queues are reference counted
  - dispatch_retain() / dispatch_release()
- GCD retains parameters to dispatch API as necessary
- Ensure correct queue lifetime across asynchronous operations

Managing Queue Lifetime

```
- (void)asyncParseData:(NSData *)data
            queue:(dispatch_queue_t)queue
            block:(void (^)(id result))block {
    dispatch_retain(queue);
    dispatch_async(self.parse_queue, ^{
        id result = [self parseData:data];
        dispatch_async(queue, ^{
            block(result);
        });
        dispatch_release(queue);
    });
}
```

Managing Object Lifetime

- Ensure objects captured by blocks are valid when blocks are executed
- Objective-C objects are auto-retained/released
- Other objects must be retained by your code
  - CFRetain() / CFRelease()

App Design with Queues

App Design with Queues

- One queue per task or subsystem
- Communicate with dispatch_async()
- Queues are lightweight and efficient
- Automatic thread recycling

App Design with Queues

Demo app tasks

1. Receive and parse network stream
2. Maintain message history
3. Fetch and cache images
4. Display user interface

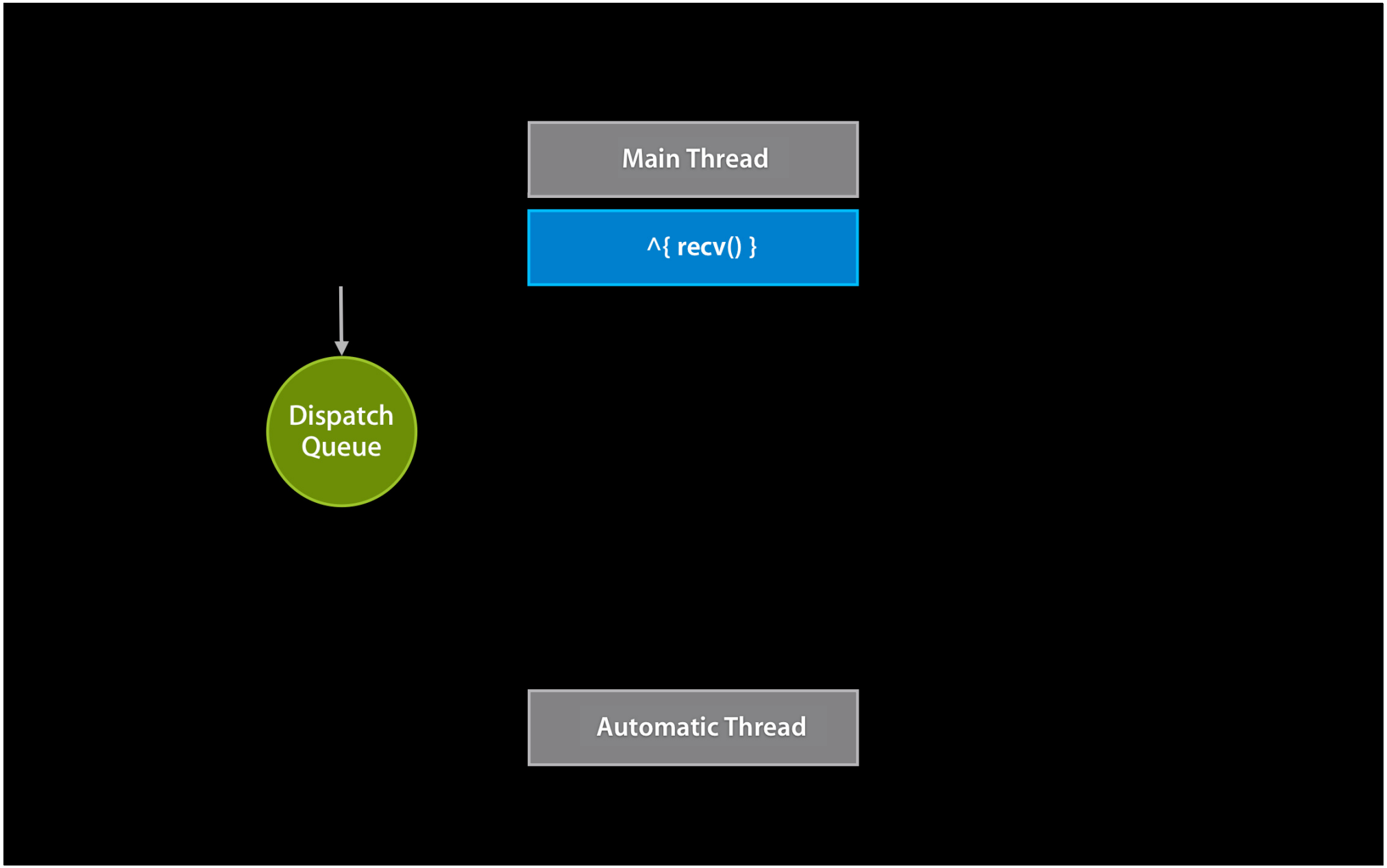
App Design with Queues

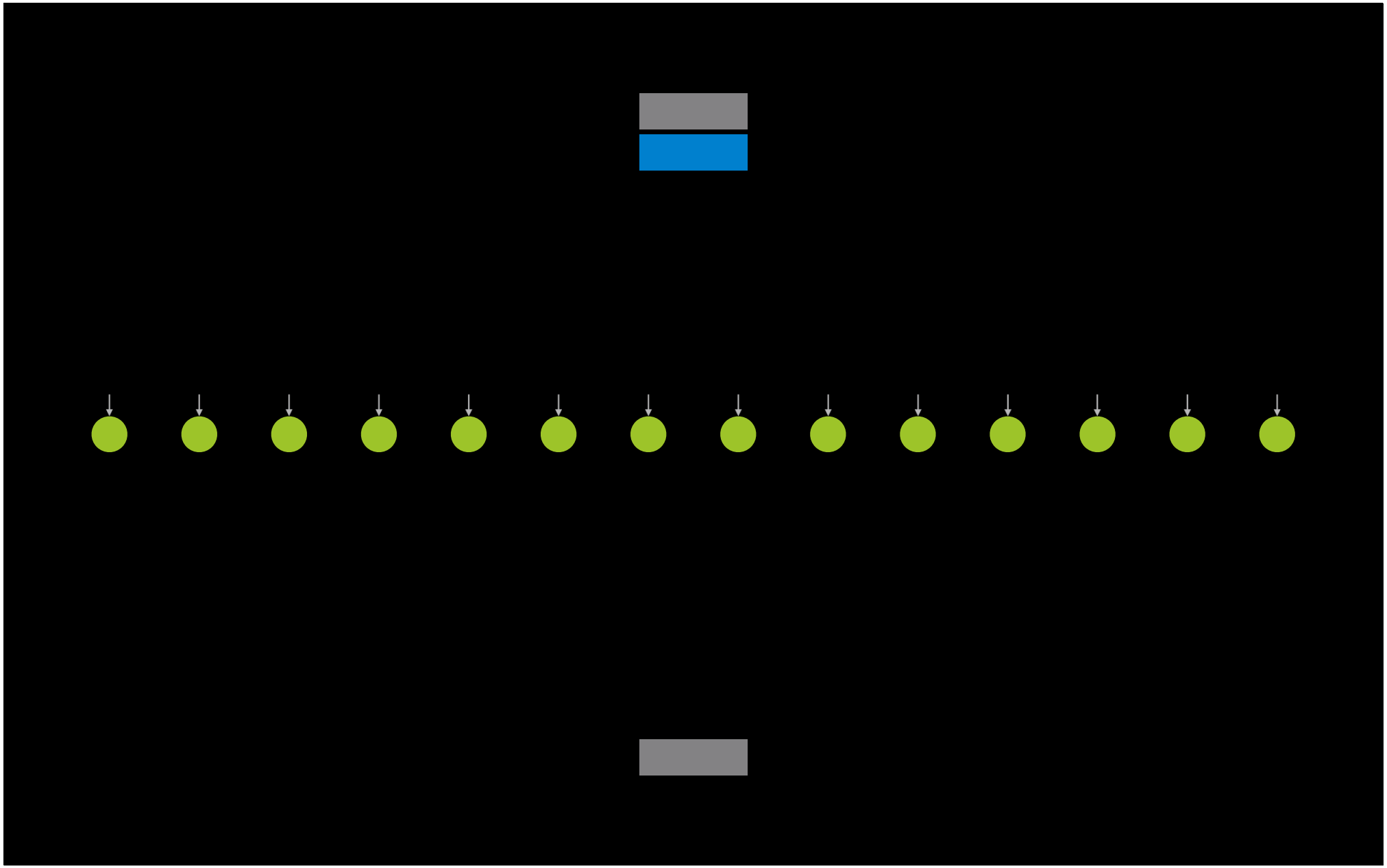
```
dispatch_async(network_queue, ^{
    NSData *d = [twitterStream receiveData];
    DTweet *tw = [[DTweet alloc] initWithData:d];
    dispatch_async(tweets_queue, ^{
    [tweets addTweet:tw];
        dispatch_async(main_queue, ^{
    [viewController displayTweet:tw];
        });
        dispatch_async(image_queue, ^{
    tw.img = [imageCache getImgFromURL:tw.url];
            dispatch_async(main_queue, ^{
    [viewController updateTweetDisplay:tw];
            });
        });
    });
    [tw release];
});
```

App Design with Queues

Pitfalls

- Avoid blocking per-subsystem queues
- Be careful when waiting
- Blocked worker threads consume resources


Main Thread
^{ recv() }
Dispatch Queue
Automatic Thread



Responding to External Events

Responding to External Events

Dispatch sources

- Monitor external events
  - Files, Network Sockets, Directories, Timers
- Event handlers can be delivered to any queue
- Use sources to replace polling or blocking API calls
- See session:
  - Simplifying iPhone App Development with Grand Central Dispatch

Responding to External Events

```
dispatch_async(network_queue, ^{
    NSData *d = [twitterStream receiveData];
    DTweet *tw = [[DTweet alloc] initWithData:d];
    dispatch_async(tweets_queue, ^{
        [tweets addTweet:tw];
        dispatch_async(main_queue, ^{
            [viewController displayTweet:tw];
        });
        dispatch_async(image_queue, ^{
            tw.img = [imageCache getImgFromURL:tw.url];
            dispatch_async(main_queue, ^{
                [viewController updateTweetDisplay:tw];
            });
        });
    });
    [tw release];
});
```

```
dispatch_source_set_event_handler(network_source, ^{
```

Where Do I Find GCD ?

Where Do I Find GCD ?

- GCD is part of libSystem.dylib
- Available to all apps

```
#include <dispatch/dispatch.h>
```

- Open Source
  - http://libdispatch.macosforge.org/
  - libdispatch-dev@lists.macosforge.org

More Information

Michael Jurewitz
Developer Tools and Performance Evangelist
jurewitz@apple.com

Documentation
Concurrency Programming Guide
http://developer.apple.com

Open Source
Mac OS Forge > libdispatch
http://libdispatch.macosforge.org

Apple Developer Forums
http://devforums.apple.com

Related Sessions

Simplifying iPhone App Development with Grand Central Dispatch	Mission Friday 10:15AM
Working Effectively with Objective-C on iPhone OS	Pacific Heights Wednesday 9:00AM
Advanced Objective-C and Garbage Collection Techniques	Pacific Heights Friday 11:30AM

Labs

Objective-C and Garbage Collection Lab	Application Frameworks Lab B Wednesday 2:00PM
Grand Central Dispatch Lab	Core OS Lab A Wednesday 4:30PM
Grand Central Dispatch Lab	Core OS Lab A Thursday 9:00AM
Grand Central Dispatch Lab	Core OS Lab A Friday 11:30AM


WWDC10