OBJECTIVE-C

THE NSHIPSTER FAKE BOOK

The NSHipster Fake Book

Mattt Thompson

Contents

Fake books are an indispensable tool for jazz musicians. They contain the melody, rhythm, and chord changes for hundreds of standards, allowing a player to jump into any session cold, and "fake it" through any tunes outside their repertoire. It's not sheet music, but rather the essence of tunes.

$C_m{}^{11}$ B^9 $Bb_m{}^9$ Ab^{13} G^{13} and trade fours with the tenor.

Programmers aren't all that different. Once you get the fundamentals down, it's really just about memorizing the APIs. Sure, you could read the docs, but nothing compares to the excitement of tinkering with code.

That's the idea behind The NSHipster Fake Book: expanding your repertoire through concise code samples, with minimal explanation. It's about letting concepts click as you discover new licks you'd never though to try. Just pick it up and start playing—no matter what kind of chops you have.

In this book, you'll find over 200 code samples, ranging from the beginner and basic to the expert and obscure, across a variety of genres and use cases.

Chapter 1

Standards

1.1 Creating a Nonretained Object Value

```
id nonCopyableObject = ...;
id <NSCopying> copyableValue = [NSValue valueWithNonretainedObject: ↩
   nonCopyableObject];
```

1.2 Adding an Anonymous Observer to a Notification Center

```
NSNotificationCenter *center = [NSNotificationCenter defaultCenter];
[center addObserverForName:nil
                    object:nil
                     queue:nil
                usingBlock:^(NSNotification *notification)
{
     NSLog(@"%@", notification.name);
}];
```

1.3 Accessing Thread-Unsafe Objects from a Thread Dictionary

```
NSMutableDictionary *threadDictionary = [[NSThread currentThread]  ↩
   threadDictionary];
NSDateFormatter *dateFormatter = threadDictionary[@"dateFormatter"];
if (!dateFormatter) {
```

```
    dateFormatter = [[NSDateFormatter alloc] init];
    dateFormatter.locale = [NSLocale currentLocale];
    dateFormatter.dateStyle = NSDateFormatterLongStyle;
    dateFormatter.timeStyle = NSDateFormatterShortStyle;
    threadDictionary[@"dateFormatter"] = dateFormatter;
}

return dateFormatter;
```

1.4 Converting a String to a Number

```
NSNumberFormatter *numberFormatter = [[NSNumberFormatter alloc] init];
NSNumber *number = [numberFormatter numberFromString:@"42"];
```

```
NSString *string = [@(42) stringValue];
```

1.5 Converting a Number to a String

```
NSNumber *number = @(42);
NSString *string =
    [NSNumberFormatter localizedStringFromNumber:number
                                     numberStyle:NSNumberFormatterNoStyle];
```

1.6 Implementing Indexed Subscripting

```
- (id)objectAtIndexedSubscript:(NSUInteger)idx;
- (void)setObject:(id)obj atIndexedSubscript:(NSUInteger)idx;
```

1.7 Implementing Keyed Subscripting

```
- (id)objectForKeyedSubscript:(id <NSCopying>)key;
- (void)setObject:(id)obj forKeyedSubscript:(id <NSCopying>)key;
```

1.8 Using a Method that accepts an Error Parameter

```
NSError *error = nil;
BOOL success =
  [[NSFileManager defaultManager] moveItemAtPath:@"/target"
                                          toPath:@"/destination"
                                           error:&error];
if (!success) {
    NSLog(@"%@", error);
}
```

1.9 Implementing a Method that takes an Error Parameter

```
- (BOOL)validateObject:(id)object
                 error:(NSError * __autoreleasing *)error
{
    BOOL success = ...;

    if (!success) {
      if (error) {
        *error = [NSError errorWithDomain:NSHipsterErrorDomain
                                     code:-42
                                 userInfo:nil];
      }
    }

    return success;
}
```

1.10 Logging Class, Method, and Line Number Context

```
NSLog(@"<%@:%@:%d>",
    NSStringFromClass([self class]),
    NSStringFromSelector(_cmd),
    __LINE__);
```

```
NSLog(@"%s", __PRETTY_FUNCTION__);
```

1.11 Key-Value Archiving & Unarchiving an Object

Archiving

```
id object = ...;
[NSKeyedArchiver archiveRootObject:object
                            toFile:@"/path/to/archive"];
```

Unarchiving

```
id object = [NSKeyedUnarchiver unarchiveObjectWithFile:@"/path/to/archive"];
```

1.12 Archiving / Unarchiving an Object into NSUserDefaults

```
NSData *data = [NSKeyedArchiver archivedDataWithRootObject:books];
[[NSUserDefaults standardUserDefaults] setObject:data forKey:@"books"];
```

```
NSData *data = [[NSUserDefaults standardUserDefaults] objectForKey:@"books" ↩
   ];
NSArray *books = [NSKeyedUnarchiver unarchiveObjectWithData:data];
```

1.13 Creating a KeyPath from Selector

```
NSString *keyPath = NSStringFromSelector(@selector(method))
```

1.14 Adding an Item to the Keychain

```
NSString *key, service;
NSData *data;

NSDictionary *query = @{
    (__bridge id)kSecClass: (__bridge id)kSecClassGenericPassword,
    (__bridge id)kSecAttrService: service,
    (__bridge id)kSecAttrGeneric: key,
    (__bridge id)kSecAttrAccount: key,
};
```

```
OSStatus status = SecItemCopyMatching((__bridge CFDictionaryRef)query, NULL) ↩
    ;

if (status == errSecSuccess) {
    NSDictionary *updatedAttributes =
        @{(__bridge id)kSecValueData: data};

    SecItemUpdate((__bridge CFDictionaryRef)query,
                  (__bridge CFDictionaryRef)updatedAttributes);
} else {
    NSMutableDictionary *attributes = [query mutableCopy];
    attributes[(__bridge id)kSecValueData] = data;
    attributes[(__bridge id)kSecAttrAccessible] =
        (__bridge id)kSecAttrAccessibleAfterFirstUnlock;

    SecItemAdd((__bridge CFDictionaryRef)attributes, NULL);
}
```

1.15 Evaluating a Mathematical Expression

```
NSExpression *expression = [NSExpression expressionWithFormat:@"4 + 5 - 2**3 ↩
    "];
id value = [expression expressionValueWithObject:nil context:nil]; // => 1
```

1.16 Decoding JSON

```
NSData *data;
NSError *error = nil;
id object = [NSJSONSerialization JSONObjectWithData:data options:0 error:& ↩
    error];
```

1.17 Encoding JSON

```
id object;
NSError *error = nil;
```

```
NSData *data = [NSJSONSerialization dataWithJSONObject:object options:0  ↩
   error:&error];
```

1.18 Getting the Name of the Device

iOS

```
[[UIDevice currentDevice] name]
```

OSX

```
SCDynamicStoreCopyComputerName(NULL, NULL);
```

Chapter 2

Language & Runtime

2.1 Declaring an NS_ENUM Type

```
typedef NS_ENUM(NSInteger, UITableViewCellStyle) {
    UITableViewCellStyleDefault,
    UITableViewCellStyleValue1,
    UITableViewCellStyleValue2,
    UITableViewCellStyleSubtitle
};
```

2.2 Declaring an NS_OPTIONS Type

```
typedef NS_OPTIONS(NSUInteger, UIViewAutoresizing) {
    UIViewAutoresizingNone                 = 0,
    UIViewAutoresizingFlexibleLeftMargin   = 1 << 0,
    UIViewAutoresizingFlexibleWidth        = 1 << 1,
    UIViewAutoresizingFlexibleRightMargin  = 1 << 2,
    UIViewAutoresizingFlexibleTopMargin    = 1 << 3,
    UIViewAutoresizingFlexibleHeight       = 1 << 4,
    UIViewAutoresizingFlexibleBottomMargin = 1 << 5
};
```

2.3 Creating String Representations for Enumerated Type

```
NSString * const UITableViewCellStyleDescription[] = {
    [UITableViewCellStyleDefault] = @"Default",
    [UITableViewCellStyleSubtitle] = @"Subtitle",
    [UITableViewCellStyleValue1] = @"Value 1",
    [UITableViewCellStyleValue2] = @"Value 2"
};
```

```
UITableViewCellStyle style = ...;
NSString *description = UITableViewCellStyleDescription[style];
```

2.4 Adding a Property to a Category

NSObject+AssociatedObject.h

```
@interface NSObject (AssociatedObject)
@property (nonatomic, strong) id associatedObject;
@end
```

NSObject+AssociatedObject.m

```
@implementation NSObject (AssociatedObject)
@dynamic associatedObject;

- (void)setAssociatedObject:(id)object {
    objc_setAssociatedObject(self, @selector(associatedObject), object, ↩
   OBJC_ASSOCIATION_RETAIN_NONATOMIC);
}

- (id)associatedObject {
    return objc_getAssociatedObject(self, @selector(associatedObject));
}
```

2.5 Swizzling a Method

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

@implementation UIViewController (Tracking)
```

```
+ (void)load {
    static dispatch_once_t onceToken;
    dispatch_once(&onceToken, ^{
        Class class = [self class];

        // When swizzling a class method, use the following:
        // Class class = object_getClass((id)self);

        SEL originalSelector = @selector(viewWillAppear:);
        SEL swizzledSelector = @selector(xxx_viewWillAppear:);

        Method originalMethod = class_getInstanceMethod(class, originalSelector);
        Method swizzledMethod = class_getInstanceMethod(class, swizzledSelector);

        BOOL didAddMethod =
            class_addMethod(class,
                originalSelector,
                method_getImplementation(swizzledMethod),
                method_getTypeEncoding(swizzledMethod));

        if (didAddMethod) {
            class_replaceMethod(class,
                swizzledSelector,
                method_getImplementation(originalMethod),
                method_getTypeEncoding(originalMethod));
        } else {
            method_exchangeImplementations(originalMethod, swizzledMethod);
        }
    });
}

#pragma mark - Method Swizzling

- (void)xxx_viewWillAppear:(BOOL)animated {
    [self xxx_viewWillAppear:animated];
    NSLog(@"viewWillAppear: %@", self);
}

@end
```

2.6 Determining the Type of a Property

```
const char *attributes = property_getAttributes(class_getProperty([self ↩
    class], sel_getName(@selector(property))));
NSString *typeAttribute = [[[NSString stringWithUTF8String:attributes] ↩
    componentsSeparatedByString:@","] firstObject];
const char *propertyType = [[typeAttribute substringFromIndex:1] UTF8String ↩
    ];

if (strcmp(propertyType, @encode(float)) == 0) {
    // float
} else if (strcmp(propertyType, @encode(int)) == 0) {
    // int
} ...
```

2.7 Determining the Type of a CFTypeRef

```
CFTypeRef ref = ...;
BOOL isCFStringRef = CFGetTypeID(ref) == CFStringGetTypeID();
```

2.8 Specifying the Availability of a Method

```
void foo() __attribute__((availability(macosx,
                                       introduced=10.4,
                                       deprecated=10.6,
                                       obsoleted=10.7)));
```

Table 2.1: Arguments

`introduced`	The first version in which this declaration was introduced.
`deprecated`	The first version in which this declaration was deprecated, meaning that users should migrate away from this API.
`obsoleted`	The first version in which this declaration was obsoleted, meaning that it was removed completely and can no longer be used.
`unavailable`	This declaration is never available on this platform.

Table 2.1: (continued)

`message`	Additional message text that Clang will provide when emitting a warning or error about use of a deprecated or obsoleted declaration. Useful to direct users to replacement APIs.

Table 2.2: Supported Platforms

`ios`	Apple's iOS operating system. The minimum deployment target is specified by the `-mios-version-min=*version*` or `-miphoneos-version-min=*version*` command-line arguments.
`macosx`	Apple's Mac OS X operating system. The minimum deployment target is specified by the `-mmacosx-version-min=*version*` command-line argument.

2.9 Hiding a Class

```
__attribute__((visibility("hidden"))
@interface HiddenClass : Superclass
// ...
@end
```

2.10 Hiding a Method

```
- (BOOL)respondsToSelector:(SEL)selector {
    if (selector == @selector(methodToHide)) {
        return NO;
    }

    return [[self class] instancesRespondToSelector:selector];
}
```

2.11 Ignoring Compiler Warnings

```
#pragma clang diagnostic push
#pragma clang diagnostic ignored "-Wgnu"
id object = object ?: [NSNull null];
#pragma clang diagnostic pop
```

2.12 Determining the Current System Memory Usage

```
#import <mach/mach.h>
#import <sys/sysctl.h>

vm_statistics_data_t vmStats;
mach_msg_type_number_t infoCount = HOST_VM_INFO_COUNT;
kern_return_t status = host_statistics(mach_host_self(), HOST_VM_INFO, ( ↩
   host_info_t)&vmStats, &infoCount);

natural_t memoryUsage = 0; // in bytes
if (status == KERN_SUCCESS) {
  memoryUsage = vmStats.wire_count * 1024.0;
}
```

2.13 Getting the Current OS Version

iOS

```
[[[UIDevice currentDevice] systemVersion]
```

OS X

```
[[NSProcessInfo processInfo] operatingSystemVersionString];
```

2.14 Declaring a Constant Value

Interface (.h File)

```
extern NSString * const XXConstant;
```

Implementation (.m File)

```
NSString * const XXConstant = @"com.example.constant"
```

2.15 Determining Whether an Object is Null

```
id object;
BOOL isNull = [object isEqual:[NSNull null]];
```

2.16 Declaring an Argument or Parameter as Unused

Function Argument

```
void foo(__unused unsigned long options) {
    // ...
}
```

Method Parameter

```
NSArray *array = @[...];
[array enumerateObjectsUsingBlock:
 ^(id object, __unused NSUInteger idx, __unused BOOL *stop) {
    NSLog(@"%@", object);
}];
```

2.17 Creating Variadic Method

```
- (void)method:(id)object, ... NS_REQUIRES_NIL_TERMINATION {
    va_list args;
    va_start(args, object);
    while (object) {
        // ...
        object = va_arg(args, id);
    }
    va_end(args);
}
```

2.18 Creating a Variadic Function

Count as First Argument

```
static double average(int count, ...) {
    va_list args;
    va_start(args, count);

    int sum = 0;
    for (int i = 0; i < count; i++) {
        sum += va_arg(args, int);
    }
    va_end(args);

    return (double)sum / (double)count;
}
```

```
double avg = average(4, 2, 3, 4, 5);
```

Sentinel Value as Last Argument

```
static double average(int x, ...) {
    va_list args;
    va_start(args, x);

    long long int sum = 0;
    int count = 0;

    do {
        x = va_arg(args, int);
        sum += x;
        count++;
    } while (x != NULL);

    va_end(args);

    return (double)sum / (double)count;
}
```

```
double avg = average(2, 3, 4, 5, NULL);
```

2.19 Overloading Functions

```
__attribute__((overloadable)) CGFloat CGFloat_floor(double d) {
    return (CGFloat)floor(d);
}

__attribute__((overloadable)) CGFloat CGFloat_floor(float f) {
    return (CGFloat)floorf(f);
}



#include <math.h>
float __attribute__((overloadable)) tgsin(float x) { return sinf(x); }
double __attribute__((overloadable)) tgsin(double x) { return sin(x); }
long double __attribute__((overloadable)) tgsin(long double x) { return sinl ↩
    (x); }
```

2.20 Determining if ARC is Available

```
#if __has_feature(objc_arc)
// ARC is Available
#endif
```

2.21 Conditionally Compiling for iOS & OS X Targets

```
#import <Availability.h>

id color = nil;

#if defined(__MAC_OS_X_VERSION_MAX_ALLOWED)
color = [NSColor orangeColor];
#elsif defined(__IPHONE_OS_VERSION_MIN_REQUIRED)
color = [UIColor purpleColor];
#endif
```

2.22 Determining if Class is Available for Target Platform at Runtime

For frameworks that support the `NS_CLASS_AVAILABLE` macro:

```
if ([NSProgress class]) {
    // NSProgress Available
} else {
    // NSProgress Unavailable
}
```

Otherwise, use `NSClassFromString`:

```
Class class = NSClassFromString(@"NSProgress");
if (class) {
    // NSProgress Available
} else {
    // NSProgress Unavailable
}
```

2.23 Determining if Method is Available for Target Platform at Runtime

Class Methods

```
SEL selector = @selector(currentLanguageCode:);
if ([AVSpeechSynthesisVoice respondToSelector:selector]) {
    // Method Available
} else {
    // Method Unavailable
}
```

Instance Methods

```
SEL selector = @selector(initWithLeaderboardIdentifier:);
if ([GKScore instancesRespondToSelector:selector]) {
    // Method Available
} else {
    // Method Unavailable
}
```

2.24 Determining if Function is Available for Target Platform at Runtime

```
if (CGColorCreateGenericGray != NULL) {
    // Function Available
} else {
    // Function Unavailable
}
```

2.25 Adding a Class at Runtime

```
Class c = objc_allocateClassPair([NSObject class], "Product", 0);
class_addIvar(c, "name", sizeof(id),
              log2(sizeof(id)), @encode(id));
class_addIvar(c, "price", sizeof(double),
              log2(sizeof(double)), @encode(double));

Ivar nameIvar = class_getInstanceVariable(c, "name");
ptrdiff_t priceIvarOffset =
    ivar_getOffset(class_getInstanceVariable(c, "price"));

IMP initIMP = imp_implementationWithBlock(
  ^(id self, NSString *name, double price) {
    object_setIvar(self, nameIvar, name);

    char *ptr = ((char *)(__bridge void *)self) + priceIvarOffset;
    memcpy(ptr, &price, sizeof(price));

    return self;
});
const char *initTypes =
    [[NSString stringWithFormat:@"%s%s%s%s%s%s",
        @encode(id), @encode(id), @encode(SEL),
        @encode(id), @encode(id), @encode(NSUInteger)] UTF8String];
class_addMethod(c, @selector(initWithFirstName:lastName:age:),
                initIMP, initTypes);

IMP nameIMP = imp_implementationWithBlock(
```

```
^(id self) {
    return object_getIvar(self, nameIvar);
});
const char *nameTypes =
    [[NSString stringWithFormat:@"%s%s%s",
        @encode(id), @encode(id), @encode(SEL)] UTF8String];
class_addMethod(c, @selector(name), nameIMP, nameTypes);

IMP priceIMP = imp_implementationWithBlock(
^(id self) {
    char *ptr = ((char *)(__bridge void *)self) + priceIvarOffset;
    double price;
    memcpy(&price, ptr, sizeof(price));

    return price;
});
const char *priceTypes =
    [[NSString stringWithFormat:@"%s%s%s",
        @encode(double), @encode(id), @encode(SEL)] UTF8String];
class_addMethod(c, @selector(age), priceIMP, priceTypes);

objc_registerClassPair(c);
```

Equivalent Class Declaration

```
@interface Product : NSObject
@property NSString *name;
@property double price;
- (instancetype)initwithName:(NSString *)name
                       price:(double)price;
@end
```

2.26 Adding a Method to a Class at Runtime

```
@interface NSObject ()
- (NSString *)greetingWithName:(NSString *)name;
@end
```

```
Class c = [NSObject class];
IMP greetingIMP = imp_implementationWithBlock(
```

```
  (NSString *)^(id self, NSString *name){
      return [NSString stringWithFormat:@"Hello, %@!", name];
});
const char *greetingTypes =
    [[NSString stringWithFormat:@"%s%s%s",
        @encode(id), @encode(id), @encode(SEL)] UTF8String];
class_addMethod(c, @selector(greetingWithName:),
                greetingIMP, greetingTypes);
```

2.27 Getting the Subclasses and Superclasses of a Class

NSObject+Hierarchy.h

```
@interface NSObject (Hierarchy)
+ (NSSet *)subclasses;
+ (NSSet *)superclasses;
@end
```

NSObject+Hierarchy.m

```
@implementation NSObject (Hierarchy)

+ (NSSet *)subclasses {
    NSMutableSet *mutableSubclasses = [NSMutableSet set];

    unsigned int count;
    Class *classes = objc_copyClassList(&count);
    for (int i = 0; i < count; i++) {
        Class class = classes[i];
        for (Class superclass = class_getSuperclass(class);
             superclass != Nil;
             superclass = class_getSuperclass(superclass))
        {
            if (superclass == self) {
                [mutableSubclasses addObject:class];
            }
        }
    }
    free(classes);

    return [NSSet setWithSet:mutableSubclasses];
```

```
}

+ (NSArray *)superclasses {
    NSMutableArray *mutableSuperclasses = [NSMutableArray array];

    for (Class superclass = class_getSuperclass(self);
         superclass != Nil;
         superclass = class_getSuperclass(superclass))
    {
        [mutableSuperclasses addObject:superclass];
    }

    return [NSArray arrayWithArray:mutableSuperclasses];
}

@end
```

2.28 Requiring Method to call super

```
- (void)method __attribute__((objc_requires_super));
```

2.29 Determining the Caller of a Method

Stack	Framework	Address	Class	Function	Line
1	UIKit	0x0047b14f	UIApplication	...	32

```
NSString *callerSymbol = [NSThread callStackSymbols][1];

NSCharacterSet *characterSet =
  [NSCharacterSet characterSetWithCharactersInString:@" +,-.?[]"];

NSArray *components =
  [callerSymbol componentsSeparatedByCharactersInSet:characterSet];
components =
  [components filteredArrayUsingPredicate:
      [NSPredicate predicateWithFormat:@"self <> ''"]];
```

```
NSString *stack = components[0];
NSString *framework = components[1];
NSString *address = components[2];
NSString *classCaller = components[3];
NSString *functionCaller = components[4];
NSString *lineCaller = components[5];
```

2.30 Creating Variadic Formatting Method

```
- (NSString *)customStringWithFormat:(NSString *)format, ... ↩
   NS_FORMAT_FUNCTION(1,0) {
    va_list args;
    va_start(args, format);
    NSString *string = [[NSString alloc] initWithFormat:format arguments: ↩
   args];
    va_end(args);

    return string;
}
```

2.31 Intentionally Crashing the Current Process

```
__builtin_trap();
```

Chapter 3

Grand Central Dispatch

3.1 Dispatching Work Asynchronously to a Background Queue

```
dispatch_queue_t backgroundQueue =
    dispatch_get_global_queue(DISPATCH_QUEUE_PRIORITY_BACKGROUND, 0);
dispatch_async(backgroundQueue, ^{
    // Do Work

    dispatch_async(dispatch_get_main_queue(), ^{
        // Return Result
    });
});
```

3.2 Benchmarking the Execution Time of an operation

```
extern uint64_t dispatch_benchmark(size_t count, void (^block)(void))...----
```

```
size_t const objectCount = 1000;
uint64_t t = dispatch_benchmark(10000, ^{
    @autoreleasepool {
        id obj = @42;
        NSMutableArray *array = [NSMutableArray array];
        for (size_t i = 0; i < objectCount; ++i) {
            [array addObject:obj];
        }
    }
```

```
});
NSLog(@"-[NSMutableArray addObject:] : %llu ns", t);
```

3.3 Monitoring Local File Changes

```
NSURL *fileURL = [[[NSFileManager defaultManager]
                    URLsForDirectory:NSDocumentDirectory
                           inDomains:NSUserDomainMask] firstObject];

dispatch_queue_t queue =
  dispatch_get_global_queue(DISPATCH_QUEUE_PRIORITY_DEFAULT, 0);

int fileDescriptor =
  open([fileURL fileSystemRepresentation], O_EVTONLY);

unsigned long mask = DISPATCH_VNODE_EXTEND |
                     DISPATCH_VNODE_WRITE  |
                     DISPATCH_VNODE_DELETE;
__block dispatch_source_t source =
    dispatch_source_create(DISPATCH_SOURCE_TYPE_VNODE,
                           fileDescriptor,
                           mask,
                           queue);

dispatch_source_set_event_handler(source, ^{
    dispatch_source_vnode_flags_t flags =
        dispatch_source_get_data(source);

    if (flags) {
        dispatch_source_cancel(source);
        dispatch_async(dispatch_get_main_queue(), ^{
            // ...
        });
    }
});

dispatch_source_set_cancel_handler(source, ^{
    close(fileDescriptor);
});
```

```
dispatch_resume(source);
```

3.4 Creating a Singleton

```
+ (instancetype)sharedInstance {
    static dispatch_once_t once;
    static id _sharedInstance = nil;
    dispatch_once(&once, ^{
        _sharedInstance = [[self alloc] init];
    });

    return _sharedInstance;
}
```

3.5 Monitoring the Parent Process PID

```
pid_t ppid = getppid();

dispatch_queue_t globalQueue =
    dispatch_get_global_queue(DISPATCH_QUEUE_PRIORITY_DEFAULT, 0);

dispatch_source_t source =
    dispatch_source_create(DISPATCH_SOURCE_TYPE_PROC,
                           ppid,
                           DISPATCH_PROC_EXIT,
                           globalQueue);
if (source) {
    dispatch_source_set_event_handler(source, ^{
        NSLog(@"pid: %d Exited", ppid);
        dispatch_source_cancel(source);
    });

    dispatch_resume(source);
}
```

3.6 Reading from STDIN

```
dispatch_queue_t globalQueue =
    dispatch_get_global_queue(DISPATCH_QUEUE_PRIORITY_DEFAULT, 0);

dispatch_source_t stdinReadSource =
    dispatch_source_create(DISPATCH_SOURCE_TYPE_READ,
                           STDIN_FILENO,
                           0,
                           globalQueue);

dispatch_source_set_event_handler(stdinReadSource, ^{
    uint8_t buffer[1024];
    int length = read(STDIN_FILENO, buffer, sizeof(buffer));
    if (length > 0) {
        NSString *string =
            [[NSString alloc] initWithBytes:buffer
                                     length:length
                                   encoding:NSUTF8StringEncoding];
        NSLog(@"%@", string);
    }
});

dispatch_resume(stdinReadSource);
```

3.7 Monitoring Local File Changes

```
NSURL *fileURL = [[[NSFileManager defaultManager]
                    URLsForDirectory:NSDocumentDirectory
                           inDomains:NSUserDomainMask] firstObject];

dispatch_queue_t queue =
  dispatch_get_global_queue(DISPATCH_QUEUE_PRIORITY_DEFAULT, 0);

int fileDescriptor =
  open([fileURL fileSystemRepresentation], O_EVTONLY);

unsigned long mask = DISPATCH_VNODE_EXTEND |
                     DISPATCH_VNODE_WRITE  |
```

```
                    DISPATCH_VNODE_DELETE;
__block dispatch_source_t source =
    dispatch_source_create(DISPATCH_SOURCE_TYPE_VNODE,
                           fileDescriptor,
                           mask,
                           queue);

dispatch_source_set_event_handler(source, ^{
    dispatch_source_vnode_flags_t flags =
        dispatch_source_get_data(source);

    if (flags) {
        dispatch_source_cancel(source);
        dispatch_async(dispatch_get_main_queue(), ^{
            // ...
        });
    }
});

dispatch_source_set_cancel_handler(source, ^{
    close(fileDescriptor);
});

dispatch_resume(source);
```

3.8 Dispatching a Timer

```
dispatch_queue_t queue =
    dispatch_queue_create(NULL, DISPATCH_QUEUE_CONCURRENT);

dispatch_source_t timer =
    dispatch_source_create(DISPATCH_SOURCE_TYPE_TIMER, 0, 0, queue);

int64_t delay = 30 * NSEC_PER_SEC;
int64_t leeway = 5 * NSEC_PER_SEC;
dispatch_source_set_timer(timer, DISPATCH_TIME_NOW, delay , leeway);

dispatch_source_set_event_handler(timer, ^{
    NSLog(@"Ding Dong!");
```

```
});

dispatch_resume(timer);
```

Chapter 4

Cryptography

4.1 Encrypting & Decrypting Using AES-128 With PBKDF2 Key

Key Generation

```
static NSData * AES128PBKDF2KeyWithPassword(
                            NSString *password,
                            NSData *salt,
                            NSError * __autoreleasing *error)
{
    NSCParameterAssert(password);
    NSCParameterAssert(salt);

    NSMutableData *mutableDerivedKey =
        [NSMutableData dataWithLength:kCCKeySizeAES128];

    CCCryptorStatus status =
        CCKeyDerivationPBKDF(kCCPBKDF2,
                             [password UTF8String],
                             [password lengthOfBytesUsingEncoding: ↩
   NSUTF8StringEncoding],
                             [salt bytes],
                             [salt length],
                             kCCPRFHmacAlgSHA1,
                             1024,
                             [mutableDerivedKey mutableBytes],
                             kCCKeySizeAES128);
```

```
    NSData *derivedKey = nil;
    if (status != kCCSuccess) {
        if (error) {
            *error = [[NSError alloc] initWithDomain:nil
                                                code:status
                                            userInfo:nil];
        }
    } else {
        derivedKey = [NSData dataWithData:mutableDerivedKey];
    }

    return derivedKey;
}
```

Encryption

```
static NSData * AES128EncryptedDataWithData(
                            NSData *data,
                            NSString *password,
                            NSData * __autoreleasing *salt,
                            NSData * __autoreleasing *initializationVector,
                            NSError * __autoreleasing *error)
{
    NSCParameterAssert(initializationVector);
    NSCParameterAssert(salt);

    uint8_t *saltBuffer = malloc(8);
    SecRandomCopyBytes(kSecRandomDefault, 8, saltBuffer);
    *salt = [NSData dataWithBytes:saltBuffer length:8];

    NSData *key = AES128PBKDF2KeyWithPassword(password, *salt, error);

    uint8_t *initializationVectorBuffer = malloc(kCCBlockSizeAES128);
    SecRandomCopyBytes(kSecRandomDefault,
                       kCCBlockSizeAES128,
                       initializationVectorBuffer);
    *initializationVector =
        [NSData dataWithBytes:initializationVector
                       length:kCCBlockSizeAES128];

    size_t size = [data length] + kCCBlockSizeAES128;
    void *buffer = malloc(size);
```

```
    size_t numberOfBytesEncrypted = 0;
    CCCryptorStatus status = CCCrypt(kCCEncrypt, kCCAlgorithmAES128,  ↩
   kCCOptionPKCS7Padding, [key bytes], [key length], [*initializationVector  ↩
   bytes], [data bytes], [data length], buffer, size, & ↩
   numberOfBytesEncrypted);

    NSData *encryptedData = nil;
    if (status != kCCSuccess) {
        if (error) {
            *error = [[NSError alloc] initWithDomain:nil
                                                code:status
                                            userInfo:nil];
        }
    } else {
        encryptedData = [[NSData alloc] initWithBytes:buffer
                                               length:numberOfBytesEncrypted ↩
   ];
    }

    return encryptedData;
}
```

Decryption

```
static NSData * AES128DecryptedDataWithData(
                          NSData *data,
                          NSString *password,
                          NSData *salt,
                          NSData *initializationVector,
                          NSError * __autoreleasing *error)
{
    NSData *key = AES128PBKDF2KeyWithPassword(password, salt, error);

    size_t size = [data length] + kCCBlockSizeAES128;
    void *buffer = malloc(size);

    size_t numberOfBytesDecrypted = 0;
    CCCryptorStatus status =
        CCCrypt(kCCDecrypt,
                kCCAlgorithmAES128,
                kCCOptionPKCS7Padding,
```

```
                [key bytes],
                [key length],
                [initializationVector bytes],
                [data bytes],
                [data length],
                buffer,
                size,
                &numberOfBytesDecrypted);

    NSData *encryptedData = nil;
    if (status != kCCSuccess) {
        if (error) {
            *error = [[NSError alloc] initWithDomain:nil
                                                code:status
                                            userInfo:nil];
        }
    } else {
        encryptedData = [[NSData alloc] initWithBytes:buffer
                                               length:numberOfBytesDecrypted ↩
   ];
    }

    return encryptedData;
}
```

4.2 Base64-Decoding Data

```
SecTransformRef transform =
    SecEncodeTransformCreate(kSecBase64Decoding, NULL);

NSData *data = [QRCode dataUsingEncoding:NSUTF8StringEncoding];
SecTransformSetAttribute(transform,
                         kSecTransformInputAttributeName,
                         (__bridge CFDataRef)data,
                         NULL);

NSData *decodedData =
    (__bridge_transfer NSData *)SecTransformExecute(transform, NULL);
```

```
CFRelease(transform);
```

4.3 Base64-Encoding Data

```
SecTransformRef transform =
    SecEncodeTransformCreate(kSecBase64Encoding, NULL);

SecTransformSetAttribute(transform,
                         kSecTransformInputAttributeName,
                         (__bridge CFDataRef)data,
                         NULL);

NSData *encodedData =
    (__bridge_transfer NSData *)SecTransformExecute(transform, NULL);

CFRelease(transform);
```

```
NSString *string = @"Lorem ipsum dolor sit amet.";
NSString *base64EncodedString = [[string dataUsingEncoding: ↩
   NSUTF8StringEncoding] base64EncodedStringWithOptions:0];
```

4.4 Calculating MD5 Digest

```
NSData *data = ...;

uint8_t output[CC_MD5_DIGEST_LENGTH];
CC_MD5(data.bytes, data.length, output);

NSData *checksum = [NSData dataWithBytes:output
                                  length:CC_MD5_DIGEST_LENGTH];
```

4.5 Calculating SHA-1 Digest

```
NSData *data = ...;

uint8_t output[CC_SHA1_DIGEST_LENGTH];
```

```
CC_SHA1(data.bytes, data.length, output);

NSData *checksum = [NSData dataWithBytes:output
                                  length:CC_SHA1_DIGEST_LENGTH];
```

4.6 Generating SHA-1 HMAC

```
NSData *data, *key;

unsigned int length = CC_SHA1_DIGEST_LENGTH;
unsigned char output[length];

CCHmac(kCCHmacAlgSHA1, key.bytes, key.length, data.bytes, data.length,  ↩
   output);
```

Chapter 5

Random

5.1 Creating a Random Integer

```
// Random int between 0 and N - 1
NSUInteger r = arc4random_uniform(N);

// Random int between 1 and N
NSUInteger r = arc4random_uniform(N) + 1;

// Random int between M and N
NSUInteger r = arc4random_uniform(N) + M;
```

5.2 Creating a Random Double

```
srand48(time(0));
double r = drand48();
```

`rand48` functions, unlike `arc4random` functions, require an initial value to be seeded before generating random numbers. The seed function, `srand48(3)`, should only be run once. === Creating a Random Color ===

```
srand48(time(0));
UIColor *color = [UIColor colorWithRed:drand48()
                                 green:drand48()
                                  blue:drand48()
                                 alpha:1.0f];
```

5.3 Creating a Random String

When operating on a known, contiguous range of Unicode characters, such as the lowercase letters (U+0061 — U+007A):

```
NSString *letter =
    [NSString stringWithFormat:@"%c", arc4random_uniform(26) + 'a'];
```

5.4 Creating a Random Date

```
NSTimeInterval timeInterval =
    (NSTimeInterval)arc4random_uniform(pow(2.0, 32.0) - 1.0);
NSDate *date = [NSDate dateWithTimeIntervalSinceReferenceDate:timeInterval];
```

Table 5.1: Convenient Powers of 2 for Time Intervals

2^0	1	= 1 Second
2^6	64	~ 1 Minute
2^12	4096	> 1 Hour
2^16	65536	> 1 Day
2^19	524288	< 1 Week
2^25	33554432	> 1 Year

5.5 Generating Random Bytes

```
NSUInteger length = 1024;

NSMutableData *mutableData =
    [NSMutableData dataWithLength:length];

OSStatus success =
    SecRandomCopyBytes(kSecRandomDefault,
                       length,
                       mutableData.mutableBytes);

__Require_noErr(success, exit);
```

5.6 Creating a UUID

```
NSUUID *UUID = [NSUUID UUID];
NSString *UUIDString = [UUID UUIDString];
```

5.7 Creating a GUID

```
NSString *GUIDString = [[NSProcessInfo processInfo] globallyUniqueString];
```

Chapter 6

Collections

6.1 Enumerating an Array

NSFastEnumeration

```
NSArray *array = ...;
for (id object in array) {
    // ...
}
```

Block Enumeration

```
[array enumerateObjectsUsingBlock:
  ^(id obj, NSUInteger idx, BOOL *stop) {
    // ...
}];
```

6.2 Enumerating a Dictionary

NSFastEnumeration

```
for (id key in [dictionary allKeys]) {
    id value = dictionary[key];

    // ...
}
```

Block Enumeration

```
[dictionary enumerateKeysAndObjectsUsingBlock:
  ^(id key, id obj, BOOL *stop) {
    // ...
}];
```

6.3 Creating a Mutable Copy of an Array

```
NSArray *array = ...; // Could be nil
NSMutableArray *mutableArray =
    [NSMutableArray arrayWithArray:array];
```

6.4 Creating a Mutable Copy of a Dictionary

```
NSDictionary *dictionary = ...; // Could be nil
NSMutableDictionary *mutableDictionary =
    [NSMutableDictionary dictionaryWithDictionary:dictionary];
```

6.5 Creating a Case-Insensitive Dictionary

```
static const void * NormalizeCaseRetain(CFAllocatorRef allocator,
                                        const void *value)
{
    if ([(__bridge id <NSObject>)value isKindOfClass:[NSString class]]) {
        value = (__bridge const void *)
                  [(__bridge NSString *)value lowercaseString];
    }

    return value;
};

CFDictionaryKeyCallBacks keyCallbacks =
    { 0, NormalizeCaseRetain, NULL, CFCopyDescription, CFEqual, NULL };

NSMutableDictionary *mutableNormalizingDictionary =
    (__bridge_transfer NSMutableDictionary *)
```

```
        CFDictionaryCreateMutable(NULL, 0, &keyCallbacks, NULL);
```

6.6 Accessing Mutable Dictionary in a Thread-Safe Manner

```
@property NSMutableDictionary *mutableDictionary;
@property dispatch_queue_t queue;

- (void)setObject:(id)object
           forKey:(id)key
{
    dispatch_barrier_async(self.queue, ^{
        self.mutableDictionary[key] = object;
    });
}
```

6.7 Reversing an Array

```
NSArray *array = ...;
NSArray *reversed = [[array reverseObjectEnumerator] allObjects];
```

6.8 Shuffling an Array

```
NSMutableArray *mutableArray = [NSMutableArray arrayWithArray:array];
NSUInteger count = [mutableArray count];
if (count > 1) {
  for (NSUInteger idx = count - 1; idx > 0; --idx) {
      NSUInteger randomIdx =
          arc4random_uniform((int32_t)(i + 1))
      [mutableArray exchangeObjectAtIndex:idx
                        withObjectAtIndex:randomIdx];
  }
}

NSArray *randomArray = [NSArray arrayWithArray:mutableArray];
```

6.9 Creating a String from an Array

```
NSArray *array = @[...];
NSString *string = [array componentsJoinedByString:@", "];
```

6.10 Filtering Objects in Array by Class

```
NSArray *mixedArray = @[@"a", @"b", @"c", @(1), @(2), @(3)];
NSPredicate *predicate = [NSPredicate predicateWithFormat:@"self ↩
   isKindOfClass: %@", [NSString class]];
NSArray *letters = [mixedArray filteredArrayUsingPredicate:predicate];
```

6.11 Computing the Sum of an Array

```
NSArray *array = @[@1, @2, @3];
NSNumber *sum = [array valueForKeyPath:@"@sum.self"];
```

6.12 Removing Duplicate Objects from an Array

Using KVC Collection Operator

```
NSArray *array = @[@"a", @"b", @"c", @"a", @"d"];
NSArray *uniqueArray =
    [array valueForKeyPath:@"@distinctUnionOfObjects.self"];
```

Using NSSet

```
NSSet *set = [NSSet setWithArray:array];
NSArray *uniqueArray = [set allObjects];
```

Using NSOrderedSet

```
NSOrderedSet *orderedSet = [NSOrderedSet orderedSetWithArray:array];
NSArray *uniqueArray = [orderedSet array];
```

Chapter 7

Linguistics & Typography

7.1 Creating a Font from TTF / OTF Data

```
CGDataProviderRef dataProviderRef =
    CGDataProviderCreateWithCFData((__bridge CFDataRef)data);
CGFontRef fontRef = CGFontCreateWithDataProvider(dataProviderRef);
CGDataProviderRelease(dataProviderRef);

CFErrorRef errorRef;
BOOL success = CTFontManagerRegisterGraphicsFont(fontRef, &errorRef);
NSString *fontName =
    (__bridge NSString *)CGFontCopyPostScriptName(fontRef);
CGFontRelease(fontRef);

if (success) {
    return [UIFont fontWithName:fontName size:[UIFont systemFontSize]];
} else {
    if (error) {
        *error = (__bridge NSError *)errorRef;
    }

    return nil;
}
```

7.2 Determining the Current Language

```
[[NSLocale preferredLanguages] firstObject];
```

7.3 Looking Up the Definition of a Word

OS X

```
@import CoreServices;

NSString *word = @"apple";
NSString *definition = (__bridge_transfer NSString *)DCSCopyTextDefinition( ↩
   NULL, (__bridge CFStringRef)word, CFRangeMake(0, [word length]));
NSLog(@"%@", definition);
```

iOS

```
NSString *word = @"apple";
if ([UIReferenceLibraryViewController dictionaryHasDefinitionForTerm:word])
{
    UIReferenceLibraryViewController *referenceLibraryViewController =
        [[UIReferenceLibraryViewController alloc] initWithTerm:@"apple"];
    [viewController presentViewController:referenceLibraryViewController
                                 animated:YES
                               completion:nil];
}
```

7.4 Applying Foreground Color to an Attributed String

```
NSString *string = @"red green blue";
NSMutableAttributedString *mutableAttributedString = [[ ↩
   NSMutableAttributedString alloc] initWithString:string];

[mutableAttributedString addAttribute:NSForegroundColorAttributeName
                                value:[UIColor redColor]
                                range:[string rangeOfString:@"red"]];

[mutableAttributedString addAttribute:NSForegroundColorAttributeName
                                value:[UIColor greenColor]
                                range:[string rangeOfString:@"green"]];
```

```
[mutableAttributedString addAttribute:NSForegroundColorAttributeName
                                value:[UIColor blueColor]
                                range:[string rangeOfString:@"blue"]];
```

7.5 Creating an Attributed String from HTML

```
NSString *HTML = ...;
NSDictionary *options =
    @{NSDocumentTypeDocumentAttribute: NSHTMLTextDocumentType};
NSError *error = nil;

NSAttributedString *attributedString =
    [[NSAttributedString alloc]
        initWithData:[HTML dataUsingEncoding:NSUTF8StringEncoding]
             options:options
  documentAttributes:nil
               error:&error];
```

7.6 Getting Characters from a Character Set

```
NSCharacterSet *characterSet = ...;

NSMutableArray *mutableCharacters = [NSMutableArray array];
for (NSUInteger plane = 0; plane <= 16; plane++) {
    if ([characterSet hasMemberInPlane:plane]) {
        for (UTF32Char character = (UTF32Char)(plane << 16);
             character < (plane + 1) << 16;
             character++)
        {
            if ([characterSet longCharacterIsMember:character]) {
                UTF32Char c = OSSwapHostToLittleInt32(character);
                NSString *characterString =
                    [[NSString alloc] initWithBytes:&c
                                             length:sizeof(UTF32Char)
                                           encoding:NSUTF32LittleEndianStringEncoding];
                [mutableCharacters addObject:characterString];
            }
```

```
        }
    }
}

NSArray *characters = [NSArray arrayWithArray:mutableCharacters];
```

7.7 Detecting Phone Number and Address from a String

```
NSError *error = nil;
NSDataDetector *detector =
    [NSDataDetector dataDetectorWithTypes:NSTextCheckingTypeAddress |
                                          NSTextCheckingTypePhoneNumber
                                  error:&error];

NSString *string = @"123 Main St. / (555) 555-5555";
[detector enumerateMatchesInString:string
                           options:kNilOptions
                             range:NSMakeRange(0, [string length])
                        usingBlock:
^(NSTextCheckingResult *result, NSMatchingFlags flags, BOOL *stop) {
  NSLog(@"Match: %@", result);
}];
```

7.8 Comparing Version Numbers

```
NSString *currentVersion = @"1.4.4";
BOOL isNewerVersion = [@"1.4.10" compare:currentVersion
                                 options:NSNumericSearch] ==
                      NSOrderedAscending;
```

7.9 Finding Proper Nouns in a String

```
NSString *string = ...;
NSMutableArray *mutableNames = [NSMutableArray array];
[string enumerateLinguisticTagsInRange:NSMakeRange(0, [string length])
                                scheme:NSLinguisticTagSchemeNameType
```

```
                                options:NSLinguisticTaggerJoinNames
                             orthography:nil
                              usingBlock:
 ^(NSString *tag, NSRange tokenRange, NSRange sentenceRange, BOOL *stop) {
     [mutableNames addObject:tag];
}];

NSArray *names = [NSArray arrayWithArray:mutableNames];
```

7.10 Calculating String Entropy

```
if (!string || [string length] == 0) {
    return 0.0f;
}

__block BOOL includesLowercaseCharacter = NO,
             includesUppercaseCharacter = NO,
             includesDecimalDigitCharacter = NO,
             includesPunctuationCharacter = NO,
             includesSymbolCharacter = NO,
             includesWhitespaceCharacter = NO,
             includesNonBaseCharacter = NO;

__block NSUInteger sizeOfCharacterSet = 0;

NSCountedSet *characterFrequency =
  [[NSCountedSet alloc] initWithCapacity:[string length]];

[string enumerateSubstringsInRange:NSMakeRange(0, [string length])
        options:NSStringEnumerationByComposedCharacterSequences
     usingBlock:
^(NSString *substring,
  NSRange substringRange,
  NSRange enclosingRange,
  BOOL *stop)
{
    {
        if (!includesLowercaseCharacter &&
              [[NSCharacterSet lowercaseLetterCharacterSet]
```

```
            characterIsMember:[substring characterAtIndex:0]])
    {
        includesLowercaseCharacter = YES;
        sizeOfCharacterSet += 26;
        goto next;
    }

    if (!includesUppercaseCharacter && [
          [NSCharacterSet uppercaseLetterCharacterSet]
            characterIsMember:[substring characterAtIndex:0]])
    {
        includesLowercaseCharacter = YES;
        sizeOfCharacterSet += 26;
        goto next;
    }

    if (!includesDecimalDigitCharacter &&
          [[NSCharacterSet decimalDigitCharacterSet]
            characterIsMember:[substring characterAtIndex:0]])
    {
        includesDecimalDigitCharacter = YES;
        sizeOfCharacterSet += 10;
        goto next;
    }

    if (!includesSymbolCharacter &&
          [[NSCharacterSet symbolCharacterSet]
            characterIsMember:[substring characterAtIndex:0]])
    {
        includesSymbolCharacter = YES;
        sizeOfCharacterSet += 10;
        goto next;
    }

    if (!includesPunctuationCharacter &&
          [[NSCharacterSet punctuationCharacterSet]
              characterIsMember:[substring characterAtIndex:0]])
    {
        includesPunctuationCharacter = YES;
        sizeOfCharacterSet += 20;
        goto next;
```

```
        }

        if (!includesWhitespaceCharacter &&
              [[NSCharacterSet whitespaceCharacterSet]
                  characterIsMember:[substring characterAtIndex:0]])
        {
            includesWhitespaceCharacter = YES;
            sizeOfCharacterSet += 1;
            goto next;
        }

        if (!includesNonBaseCharacter &&
              [[NSCharacterSet nonBaseCharacterSet]
                  characterIsMember:[substring characterAtIndex:0]])
        {
            includesNonBaseCharacter = YES;
            sizeOfCharacterSet += 32 + 128;
            goto next;
        }
    }
    next: {
        [characterFrequency addObject:substring];
    }
}];

CGFloat entropyPerCharacter = log2f(sizeOfCharacterSet);

CGFloat stringEntropy = entropyPerCharacter * [string length];
```

7.11 Formatting Strings

```
NSString *string =
  [NSString stringWithFormat:@"%@ %g", @"pi", M_PI];
```

Table 7.1: Type Specifiers

%@	Objective-C object, printed as the string returned by descriptionWithLocale: if available, or description otherwise. Also works with CFTypeRef objects, returning the result of the CFCopyDescription function.

Table 7.1: (continued)

%%	% character.
%d, %D	Signed 32-bit integer (int).
%u, %U	Unsigned 32-bit integer (unsigned int).
%x	Unsigned 32-bit integer (unsigned int), printed in hexadecimal using the digits 0–9 and lowercase a–f.
%X	Unsigned 32-bit integer (unsigned int), printed in hexadecimal using the digits 0–9 and uppercase A–F.
%o, %O	Unsigned 32-bit integer (unsigned int), printed in octal.
%f	64-bit floating-point number (double).
%e	64-bit floating-point number (double), printed in scientific notation using a lowercase e to introduce the exponent.
%E	64-bit floating-point number (double), printed in scientific notation using an uppercase E to introduce the exponent.
%g	64-bit floating-point number (double), printed in the style of %e if the exponent is less than –4 or greater than or equal to the precision, in the style of %f otherwise.
%G	64-bit floating-point number (double), printed in the style of %E if the exponent is less than –4 or greater than or equal to the precision, in the style of %f otherwise.
%c	8-bit unsigned character (unsigned char), printed by NSLog() as an ASCII character, or, if not an ASCII character, in the octal format \\ddd or the Unicode hexadecimal format \\udddd, where d is a digit.
%C	16-bit Unicode character (unichar), printed by NSLog() as an ASCII character, or, if not an ASCII character, in the octal format \\ddd or the Unicode hexadecimal format \\udddd, where d is a digit.
%s	Null-terminated array of 8-bit unsigned characters. Because the %s specifier causes the characters to be interpreted in the system default encoding, the results can be variable, especially with right-to-left languages. For example, with RTL, %s inserts direction markers when the characters are not strongly directional. For this reason, it's best to avoid %s and specify encodings explicitly.
%S	Null-terminated array of 16-bit Unicode characters.
%p	Void pointer (void *), printed in hexadecimal with the digits 0–9 and lowercase a–f, with a leading 0x.
%a	64-bit floating-point number (double), printed in scientific notation with a leading 0x and one hexadecimal digit before the decimal point using a lowercase p to introduce the exponent.
%A	64-bit floating-point number (double), printed in scientific notation with a leading 0X and one hexadecimal digit before the decimal point using a uppercase P to introduce the exponent.
%F	64-bit floating-point number (double), printed in decimal notation.

Table 7.2: Length Modifiers

h	Length modifier specifying that a following d, o, u, x, or X conversion specifier applies to a short or unsigned short argument.
hh	Length modifier specifying that a following d, o, u, x, or X conversion specifier applies to a signed char or unsigned char argument.
l	Length modifier specifying that a following d, o, u, x, or X conversion specifier applies to a long or unsigned long argument.
ll, q	Length modifiers specifying that a following d, o, u, x, or X conversion specifier applies to a long long or unsigned long long argument.
L	Length modifier specifying that a following a, A, e, E, f, F, g, or G conversion specifier applies to a long double argument.
z	Length modifier specifying that a following d, o, u, x, or X conversion specifier applies to a size_t or the corresponding signed integer type argument.
t	Length modifier specifying that a following d, o, u, x, or X conversion specifier applies to a ptrdiff_t or the corresponding unsigned integer type argument.
j	Length modifier specifying that a following d, o, u, x, or X conversion specifier applies to a intmax_t or uintmax_t argument.

7.12 Determining the Language of a String

```
NSString *string = @"Les Cousins Dangereux"
NSString *language =
  (__bridge NSString *)CFStringTokenizerCopyBestStringLanguage(
      (__bridge CFStringRef)string,
      CFRangeMake(0, [string length]));
```

7.13 Concatenating String Literals

```
NSString *string = @"One" @"Two" @"Three";
NSString *equivalent = @"OneTwoThree";
```

7.14 Determine if a String Contains Punctuation

```
NSString *string = ...;
NSCharacterSet *characterSet = [NSCharacterSet punctuationCharacterSet];
BOOL containsPunctuation =
    [string rangeOfCharacterFromSet:characterSet].location != NSNotFound;
```

7.15 Splitting a String into an Array

```
NSString *string = @"a,b,c,d";
NSArray *array = [string componentsSeparatedByString:@","];
```

7.16 Determining if a String Contains a Particular Substring

Case Sensitive

```
NSString *string = ...;
NSString *substring = ...;

BOOL containsSubstring =
    [string rangeOfString:substring].location != NSNotFound;
```

Case Insensitive

```
BOOL containsSubstring =
    [string rangeOfString:substring
                  options:NSCaseInsensitiveSearch].location != NSNotFound;
```

7.17 Tokenizing a String

```
[string enumerateSubstringsInRange:NSMakeRange(0, [string length])
                           options:NSStringEnumerationByWords
                        usingBlock:
 ^(NSString *word, NSRange wordRange, NSRange enclosingRange, BOOL *stop) {
     // ...
}];
```

7.18 Squashing Whitespace in a String

```
NSString *string = @"Lorem    ipsum dolar   sit  amet.";
string = [string stringByTrimmingCharactersInSet:
            [NSCharacterSet whitespaceCharacterSet]];

NSArray *components =
    [string componentsSeparatedByCharactersInSet:
        [NSCharacterSet whitespaceCharacterSet]];
components = [components filteredArrayUsingPredicate:
                [NSPredicate predicateWithFormat:@"self <> ''"]];

string = [components componentsJoinedByString:@" "];
```

7.19 Stripping Whitespace from a String

```
NSString *string = @"   asdfads ";
[string stringByTrimmingCharactersInSet:
    [NSCharacterSet whitespaceAndNewlineCharacterSet]];
```

7.20 Getting the Small Caps Variant of a Font

```
UIFont *font;

NSArray *features =
  (__bridge_transfer NSArray *)CTFontCopyFeatures((__bridge CTFontRef)font);

BOOL hasSmallCaps = NO;
for (NSDictionary *feature in features) {
    for (id selector in feature[@"CTFeatureTypeSelectors"]) {
        id name = selector[@"CTFeatureSelectorName"];

        if ([name isEqualToString:@"Small Capitals"]) {
            hasSmallCaps = YES;
            break;
        }
    }
```

```
}

if (hasSmallCaps) {
  NSArray *settings = @[
    @{
      UIFontFeatureTypeIdentifierKey : @(kLowerCaseType),
      UIFontFeatureSelectorIdentifierKey : @(kLowerCaseSmallCapsSelector)
    }
  ];

  UIFontDescriptor *descriptor =
    [font.fontDescriptor
        fontDescriptorByAddingAttributes:
            @{UIFontDescriptorFeatureSettingsAttribute : settings}];

  UIFont *smallCapsFont =
    [UIFont fontWithDescriptor:descriptor size:font.pointSize];
}
```

7.21 Responding to Changes in a Text Field

```
UITextField *textField = ...;
[textField addTarget:self selector:@selector(textFieldDidChangeText:) ]
```

7.22 Determining if a String is Empty

```
NSString *string = ...;
BOOL isEmpty = [string length] == 0;
```

```
NSString *string = ...;
BOOL isEmpty =
  [[string stringByTrimmingCharactersInSet:[NSCharacterSet ↩
   whitespaceCharacterSet]] length] == 0;
```

7.23 Determining if an Array is Empty

```
NSArray *array = ...;
BOOL isEmpty = [array count] == 0;
```

7.24 Determining if a Dictionary is Empty

```
NSDictionary *dictionary = ...;
BOOL isEmpty = [dictionary count] == 0;
```

Chapter 8

Date & Time

8.1 Extracting Components from a Date

```
NSCalendar *calendar = [NSCalendar currentCalendar];
NSDate *date = [NSDate date];

NSUInteger units = NSDayCalendarUnit | NSMonthCalendarUnit;
NSDateComponents *components = [calendar components:units
                                           fromDate:date];
```

8.2 Creating a Date from Components

```
NSCalendar *calendar = [NSCalendar currentCalendar];

NSDateComponents *components = [[NSDateComponents alloc] init];
[components setYear:1987];
[components setMonth:3];
[components setDay:17];
[components setHour:14];
[components setMinute:20];
[components setSecond:0];

NSDate *date = [calendar dateFromComponents:components];
```

8.3 Performing Calendar Arithmetic

```
NSCalendar *calendar = [NSCalendar currentCalendar];

NSDateComponents *components = [[NSDateComponents alloc] init];
[components setWeek:1];
[components setHour:12];

NSDate *oneWeekAnd12HoursFromNow =
    [calendar dateByAddingComponents:components
                              toDate:[NSDate date]
                             options:0];
```

8.4 Parsing & Formatting an ISO 8601 timestamp

```
NSDateFormatter *ISO8601DateFormatter = [[NSDateFormatter alloc] init];
NSLocale *en_US_POSIXLocale =
    [[NSLocale alloc] initWithLocaleIdentifier:@"en_US_POSIX"];

[ISO8601DateFormatter setLocale:en_US_POSIXLocale];
[ISO8601DateFormatter setDateFormat:@"yyyy-MM-dd'T'HH:mm"];
[ISO8601DateFormatter setTimeZone:[NSTimeZone timeZoneForSecondsFromGMT:0]];

NSString *timestamp = [ISO8601DateFormatter stringFromDate:[NSDate date]];
NSDate *date = [ISO8601DateFormatter dateFromString:timestamp];
```

Faster, more efficient, more flexible:

```
#include <time.h>
#include <xlocale.h>

static unsigned int const ISO_8601_MAX_LENGTH = 29;

const char *source = [timestamp cStringUsingEncoding:NSUTF8StringEncoding];
char destination[ISO_8601_MAX_LENGTH];
size_t length = strlen(source);

if (length == 0) {
    return nil;
```

```
}

double milliseconds = 0.f;
if (length == 20 && source[length - 1] == 'Z') {
    memcpy(destination, source, length - 1);
    strncpy(destination + length - 1, "+0000\0", 6);
} else if (length == 24 && source[length - 5] == '.' &&
           source[length - 1] == 'Z') {
    memcpy(destination, source, length - 5);
    strncpy(destination + length - 5, "+0000\0", 6);
    milliseconds =
        [[timestamp substringWithRange:NSMakeRange(20, 3)] doubleValue] /
            1000.f;
} else if (length == 25 && source[22] == ':') {
    memcpy(destination, source, 22);
    memcpy(destination + 22, source + 23, 2);
} else if (length == 29 && source[26] == ':') {
    memcpy(destination, source, 26);
    memcpy(destination + 26, source + 27, 2);
} else {
    memcpy(destination, source, MIN(length, ISO_8601_MAX_LENGTH - 1));
}

destination[sizeof(destination) - 1] = 0;

struct tm time = {
    .tm_isdst = -1,
};

strptime_l(destination, "%FT%T%z", &time, NULL);

NSDate *date =
    [NSDate dateWithTimeIntervalSince1970:mktime(&time) + milliseconds];
```

8.5 Parsing & Formatting an RFC 3339 Timestamp

```
NSDateFormatter *RFC3339DateFormatter = [[NSDateFormatter alloc] init];
NSLocale *en_US_POSIXLocale =
    [[NSLocale alloc] initWithLocaleIdentifier:@"en_US_POSIX"];
```

```
[RFC3339DateFormatter setLocale:en_US_POSIXLocale];
[RFC3339DateFormatter setDateFormat:@"yyyy'-'MM'-'dd'T'HH':'mm':'ss'Z'"];
[RFC3339DateFormatter setTimeZone:[NSTimeZone timeZoneForSecondsFromGMT:0]];

NSString *timestamp = [RFC3339DateFormatter stringFromDate:[NSDate date]];
NSDate *date = [RFC3339DateFormatter dateFromString:timestamp];
```

8.6 Determining if Device is Set for 24 Hour Time

```
NSString *format =
    [NSDateFormatter dateFormatFromTemplate:@"j"
                                    options:0
                                     locale:[NSLocale currentLocale]];

BOOL uses24HourTime = [format rangeOfString:@"a"].location == NSNotFound;
```

Chapter 9

Filesystem

9.1 Finding Application Support Directory

```
[[[NSFileManager defaultManager]
    URLsForDirectory:NSApplicationSupportDirectory
           inDomains:NSUserDomainMask] firstObject];
```

9.2 Finding Caches Directory

```
[[[NSFileManager defaultManager]
    URLsForDirectory:NSCachesDirectory
           inDomains:NSUserDomainMask] firstObject];
```

9.3 Listing all files in a directory

```
NSFileManager *fileManager = [NSFileManager defaultManager];
NSURL *bundleURL = [[NSBundle mainBundle] bundleURL];
NSArray *contents = [fileManager contentsOfDirectoryAtURL:bundleURL
                                  includingPropertiesForKeys:@[]
                                                     options: ↩
   NSDirectoryEnumerationSkipsHiddenFiles
                                                       error:nil];

NSPredicate *predicate = [NSPredicate predicateWithFormat:@"pathExtension == ↩
    'png'"];
```

```
for (NSURL *fileURL in [contents filteredArrayUsingPredicate:predicate]) {
    // Enumerate each .png file in directory
}
```

9.4 Creating a Directory

```
NSFileManager *fileManager = [NSFileManager defaultManager];

NSURL *documentsURL =
    [[[NSFileManager defaultManager]
      URLsForDirectory:NSDocumentDirectory
             inDomains:NSUserDomainMask] firstObject];

NSURL *imagesURL = [documentsURL URLByAppendingPathComponent:@"images"];

if (![fileManager fileExistsAtPath:[imagesURL path]]) {
    [fileManager createDirectoryAtURL:imagesURL
          withIntermediateDirectories:NO
                           attributes:nil
                                error:nil];
}
```

9.5 Recursively Enumerating all files in a directory

```
NSFileManager *fileManager = [NSFileManager defaultManager];
NSURL *bundleURL = [[NSBundle mainBundle] bundleURL];
NSDirectoryEnumerator *enumerator =
    [fileManager enumeratorAtURL:bundleURL
      includingPropertiesForKeys:@[NSURLNameKey, NSURLIsDirectoryKey]
                         options:NSDirectoryEnumerationSkipsHiddenFiles
                    errorHandler:
  ^BOOL(NSURL *url, NSError *error)
{
    NSLog(@"[Error] %@ (%@)", error, url);
}];

NSMutableArray *mutableFileURLs = [NSMutableArray array];
for (NSURL *fileURL in enumerator) {
```

```
    NSString *filename;
    [fileURL getResourceValue:&filename
                       forKey:NSURLNameKey
                        error:nil];

    NSNumber *isDirectory;
    [fileURL getResourceValue:&isDirectory
                       forKey:NSURLIsDirectoryKey
                        error:nil];

    if (![isDirectory boolValue]) {
        [mutableFileURLs addObject:fileURL];
    }
}
```

9.6 Finding Documents Directory

```
[[[NSFileManager defaultManager]
    URLsForDirectory:NSDocumentDirectory
           inDomains:NSUserDomainMask] firstObject];
```

9.7 Finding Downloads Directory

```
[[[NSFileManager defaultManager]
    URLsForDirectory:NSDownloadsDirectory
           inDomains:NSUserDomainMask] firstObject];
```

9.8 Determining the Creation Date of a File

```
NSFileManager *fileManager = [NSFileManager defaultManager];
NSURL *documentsURL =
    [[[NSFileManager defaultManager]
      URLsForDirectory:NSDocumentDirectory
      inDomains:NSUserDomainMask] firstObject];
NSURL *fileURL =
    [documentsURL URLByAppendingPathComponent:@"Document.pages"];
```

```
NSDate *creationDate = nil;
if ([fileManager fileExistsAtPath:[fileURL path]]) {
    NSDictionary *attributes =
        [fileManager attributesOfItemAtPath:filePath error:nil];
    creationDate = attributes[NSFileCreationDate];
}
```

9.9 Deleting a File

```
NSFileManager *fileManager = [NSFileManager defaultManager];
NSURL *documentsURL =
    [[[NSFileManager defaultManager]
      URLsForDirectory:NSDocumentDirectory
      inDomains:NSUserDomainMask] firstObject];
NSURL *fileURL =
    [documentsURL URLByAppendingPathComponent:@"image.png"];
NSError *error = nil;

if (![fileManager removeItemAtPath:[fileURL path]
                             error:&error])
{
    NSLog(@"[Error] %@ (%@)", error, filePath);
}
```

9.10 Determining if a File Exists

```
NSFileManager *fileManager = [NSFileManager defaultManager];
NSURL *documentsURL =
    [[[NSFileManager defaultManager]
      URLsForDirectory:NSDocumentDirectory
             inDomains:NSUserDomainMask] firstObject];
NSURL *fileURL = [documentsPath URLByAppendingPathComponent:@"file.txt"];
BOOL fileExists = [fileManager fileExistsAtPath:[fileURL path]];
```

9.11 Finding Library Directory

```
[[[NSFileManager defaultManager]
    URLsForDirectory:NSLibraryDirectory
           inDomains:NSUserDomainMask] firstObject];
```

9.12 Writing a String to Disk

```
NSURL *fileURL = [NSURL fileURLWithPath:@"/path/to/file.txt"];
NSString *string = @"Hello, World!";
NSError *error = nil;

[string writeToURL:fileURL
        atomically:YES
          encoding:NSUTF8StringEncoding
             error:&error];
```

9.13 Sorting Files by Filename

```
NSArray *fileNames = @[@"Untitled 3", @"Untitled 17", @"Untitled 5"];
NSArray *sortedFilenames = [fileNames sortedArrayUsingComparator:
^NSComparisonResult(id obj1, id obj2) {
    return [obj1 compare:obj2 options:NSNumericSearch];
}];
```

9.14 Setting Extended File Attributes

```
#include <sys/xattr.h>

NSHTTPURLResponse *response = ...;
NSURL *fileURL = ...;

const char *filePath = [fileURL fileSystemRepresentation];
const char *name = "com.Example.Etag";
const char *value = [[response allHeaderFields][@"Etag"] UTF8String];
int result = setxattr(filePath, name, value, strlen(value), 0, 0);
```

Chapter 10

Cartography

10.1 Getting Directions Between Two Locations

```
MKPlacemark *placemark = ...;

MKDirectionsRequest *request = [[MKDirectionsRequest alloc] init];
request.source = [MKMapItem mapItemForCurrentLocation];
request.destination = [[MKMapItem alloc] initWithPlacemark:placemark];

MKDirections *directions = [[MKDirections alloc] initWithRequest:request];
[directions calculateDirectionsWithCompletionHandler:
^(MKDirectionsResponse *response, NSError *error) {
    if (!error) {
        // ...
    }
}];
```

10.2 Formatting a Localized Distance String

```
CLLocation *sanFrancisco = [[CLLocation alloc] initWithLatitude:37.775 ↩
   longitude:-122.4183333];
CLLocation *portland = [[CLLocation alloc] initWithLatitude:45.5236111 ↩
   longitude:-122.675];
CLLocationDistance distance = [portland distanceFromLocation:sanFrancisco];

MKDistanceFormatter *formatter = [[MKDistanceFormatter alloc] init];
```

```
formatter.units = MKDistanceFormatterUnitsImperial;
NSString *string = [formatter stringFromDistance:distance];
```

10.3 Searching for a Geographic Location

```
MKLocalSearchRequest *request = [[MKLocalSearchRequest alloc] init];
request.region = ...;
request.naturalLanguageQuery = @"Golden Gate Bridge";

MKLocalSearch *search = [[MKLocalSearch alloc] initWithRequest:request];
[search startWithCompletionHandler:
^(MKLocalSearchResponse *response, NSError *error) {
    // ...
}];
```

10.4 Creating Map Snapshots of Waypoints

```
MKMapSnapshotOptions *options = [[MKMapSnapshotOptions alloc] init];
options.region = self.mapView.region;
options.size = self.mapView.frame.size;
options.scale = [[UIScreen mainScreen] scale];

NSURL *fileURL = [NSURL fileURLWithPath:@"path/to/snapshot.png"];

MKMapSnapshotter *snapshotter = [[MKMapSnapshotter alloc] initWithOptions: ↩
   options];
[snapshotter startWithCompletionHandler:^(MKMapSnapshot *snapshot, NSError * ↩
   error) {
    if (error) {
        NSLog(@"[Error] %@", error);
        return;
    }

    UIImage *image = snapshot.image;
    NSData *data = UIImagePNGRepresentation(image);
    [data writeToURL:fileURL atomically:YES];
}];
```

```
dispatch_queue_t globalQueue =
    dispatch_get_global_queue(DISPATCH_QUEUE_PRIORITY_DEFAULT, 0);
[snapshotter startWithQueue:globalQueue
          completionHandler:
  ^(MKMapSnapshot *snapshot, NSError *error)
{
      if (error) {
          NSLog(@"[Error] %@", error);
          return;
      }

      MKAnnotationView *pin =
          [[MKPinAnnotationView alloc] initWithAnnotation:nil
                                          reuseIdentifier:nil];

      UIImage *image = snapshot.image;
      UIGraphicsBeginImageContextWithOptions(image.size, YES, image.scale);
      {
          [image drawAtPoint:CGPointMake(0.0f, 0.0f)];

          CGRect rect = CGRectMake(0.0f, 0.0f, image.size.width, image.size. ↩
   height);
          for (id <MKAnnotation> annotation in self.mapView.annotations)
          {
              CGPoint point =
                  [snapshot pointForCoordinate:annotation.coordinate];
              if (CGRectContainsPoint(rect, point)) {
                  point.x = point.x + pin.centerOffset.x -
                                (pin.bounds.size.width / 2.0f);
                  point.y = point.y + pin.centerOffset.y -
                                (pin.bounds.size.height / 2.0f);
                  [pin.image drawAtPoint:point];
              }
          }

          UIImage *compositeImage =
              UIGraphicsGetImageFromCurrentImageContext();
          NSData *data = UIImagePNGRepresentation(compositeImage);
          [data writeToURL:fileURL atomically:YES];
      }
```

```
      UIGraphicsEndImageContext();
}];
```

10.5 Rendering Annotations on a Map View

```
- (MKAnnotationView *)mapView:(MKMapView *)mapView
            viewForAnnotation:(id <MKAnnotation>)annotation
{
    static NSString * PinIdentifier = @"Pin";

    MKAnnotationView *annotationView =
        [mapView dequeueReusableAnnotationViewWithIdentifier:PinIdentifier];

    if (!annotationView) {
        annotationView =
          [[MKPinAnnotationView alloc] initWithAnnotation:annotation
                                          reuseIdentifier:PinIdentifier];
    };

    annotationView.hidden =
        ![annotation isKindOfClass:[MKPointAnnotation class]];

    return annotationView;
}
```

10.6 Fitting a Map View to Annotations

```
NSArray *annotations = @[...];
MKMapView *mapView = ...;

[mapView showAnnotations:annotations animated:YES];
```

10.7 Rendering Overlays on an MKMapView

```
- (MKOverlayRenderer *)mapView:(MKMapView *)mapView
            rendererForOverlay:(id <MKOverlay>)overlay
```

```
{
    MKOverlayRenderer *renderer = nil;
    if ([overlay isKindOfClass:[MKPolyline class]]) {
        renderer = [[MKPolylineRenderer alloc] initWithPolyline:(MKPolyline ↩
   *)overlay];
        ((MKPolylineRenderer *)renderer).strokeColor = [UIColor greenColor];
        ((MKPolylineRenderer *)renderer).lineWidth = 3.0f;
    } else if ([overlay isKindOfClass:[MKPolygon class]]) {
        renderer = [[MKPolygonRenderer alloc] initWithPolygon:(MKPolygon *) ↩
   overlay];
        ((MKPolygonRenderer *)renderer).strokeColor = [UIColor redColor];
        ((MKPolygonRenderer *)renderer).fillColor = [UIColor colorWithRed ↩
   :1.0f green:0.0f blue:0.0f alpha:0.5f];
        ((MKPolygonRenderer *)renderer).lineWidth = 3.0f;
    }

    renderer.alpha = 0.5;

    return renderer;
}
```

```
#pragma mark - MKMapViewDelegate

- (MKOverlayRenderer *)mapView:(MKMapView *)mapView
            rendererForOverlay:(id <MKOverlay>)overlay
{
    if ([overlay isKindOfClass:[MKTileOverlay class]]) {
        return [[MKTileOverlayRenderer alloc] initWithTileOverlay:overlay];
    }

    return nil;
}
```

10.8 Localizing Address Format of a Placemark

```
MKPlacemark *placemark = ...;
BOOL includeCountryName = YES;
NSString *localizedAddress =
    ABCreateStringWithAddressDictionary(placemark, includeCountryName);
```

10.9 Describing a Placemark for a Coordinate Region

```
@interface MKPlacemark (_)
- (NSString *)descriptionForCoordinateRegion:(MKCoordinateRegion)region;
@end

@implementation MKPlacemark (_)

static CLLocationDistance const ThoroughfareDistanceThreshold = 128;
static CLLocationDistance const SubLocalityDistanceThreshold = 1024;
static CLLocationDistance const LocalityDistanceThreshold = 4096;
static CLLocationDistance const AdministrativeAreaDistanceThreshold = 65536;

- (NSString *)descriptionForCoordinateRegion:(MKCoordinateRegion)region {
    CLLocationCoordinate2D southwestCoordinate =
        CLLocationCoordinate2DMake(
            region.center.latitude - region.span.latitudeDelta,
            region.center.longitude - region.span.longitudeDelta);

    CLLocationCoordinate2D northeastCoordinate =
        CLLocationCoordinate2DMake(
            region.center.latitude + region.span.latitudeDelta,
            region.center.longitude + region.span.longitudeDelta);

    CLLocationDistance distance =
        MKMetersBetweenMapPoints(
            MKMapPointForCoordinate(southwestCoordinate),
            MKMapPointForCoordinate(northeastCoordinate));

    NSArray *addressComponentKeyPaths = @[
        NSStringFromSelector(@selector(thoroughfare)),
        NSStringFromSelector(@selector(subLocality)),
        NSStringFromSelector(@selector(subAdministrativeArea)),
        NSStringFromSelector(@selector(administrativeArea)),
        NSStringFromSelector(@selector(country))
    ];

    NSUInteger location = 0;
    if (distance > AdministrativeAreaDistanceThreshold) {
        location = 4;
```

```
    } else if (distance > LocalityDistanceThreshold) {
        location = 3;
    } else if (distance > SubLocalityDistanceThreshold) {
        location = 2;
    } else if (distance > ThoroughfareDistanceThreshold) {
        location = 1;
    }

    NSRange candidateRange =
        NSMakeRange(location, [addressComponentKeyPaths count] - location);
    for (NSString *keyPath in [addressComponentKeyPaths subarrayWithRange: ↩
   candidateRange]) {
        id value = [self valueForKeyPath:keyPath];
        if (value) {
            return value;
        }
    }

    return nil;
}

@end
```

10.10 Creating a Custom Map Tile Overlay

```
@interface XXTileOverlay : MKTileOverlay
@property NSCache *cache;
@property NSOperationQueue *operationQueue;
@end

@implementation XXTileOverlay

- (NSURL *)URLForTilePath:(MKTileOverlayPath)path {
    return [NSURL URLWithString:
        [NSString stringWithFormat:@"http://tile.example.com/%d/%d/%d",
            path.z, path.x, path.y]];
}

- (void)loadTileAtPath:(MKTileOverlayPath)path
```

```
                result:(void (^)(NSData *data, NSError *error))result
{
    if (!result) {
        return;
    }

    NSData *cachedData =
        [self.cache objectForKey:[self URLForTilePath:path]];
    if (cachedData) {
        result(cachedData, nil);
    } else {
        NSURLRequest *request =
            [NSURLRequest requestWithURL:[self URLForTilePath:path]];
        [NSURLConnection sendAsynchronousRequest:request
                                           queue:self.operationQueue
                               completionHandler:
        ^(NSURLResponse *response,
          NSData *data,
          NSError *connectionError)
        {
            result(data, connectionError);
        }];
    }
}

@end
```

10.11 Converting Degrees to Radians

```
double radians = degrees * M_PI / 180.0f;
```

10.12 Converting Radians to Degrees

```
double degrees = radians * 180.0f / M_PI;
```

10.13 Convert Radians to CLLocationDirection

```
CLLocationDirection direction = fmod(degrees, 360.0f) + 90.0f;
```

Chapter 11

Graphics

11.1 Animating a CAGradientLayer

```
NSArray *colors = @[(id)[[UIColor redColor] CGColor],
                    (id)[[UIColor orangeColor] CGColor]];
NSArray *locations = @[@(0.0), @(1.0)];
NSTimeInterval duration = 1.0f;
[UIView animateWithDuration:duration animations:^{
    [CATransaction begin];
    {
        [CATransaction setAnimationDuration:duration];
        [CATransaction setAnimationTimingFunction:[CAMediaTimingFunction ↩
   functionWithName:kCAMediaTimingFunctionEaseInEaseOut]];
        [(CAGradientLayer *)self.layer setColors:colors];
        [(CAGradientLayer *)self.layer setLocations:locations];
    }
    [CATransaction commit];
}];


+ (Class)layerClass {
    return [CAGradientLayer class];
}

#pragma mark - CALayerDelegate

- (id <CAAction>)actionForLayer:(CALayer *)layer
                         forKey:(NSString *)event
```

```
{
    id <CAAction> action = [super actionForLayer:layer forKey:event];
    if ((!action || [(id)action isEqual:[NSNull null]]) &&
        [event isEqualToString:@"colors"])
    {
        action = [CABasicAnimation animationWithKeyPath:event];
    }

    return action;
}
```

11.2 Splitting a CGRect

```
CGRect slice, remainder;
CGRectDivide(self.view.bounds, &slice, &areaTwo, 200.0f, CGRectMinXEdge);
```

11.3 Ensuring that a CGRect is not on a Pixel Boundary

```
CGRect blurryRect = CGRectMake(7.25f, 5.25f, 100.0f, 21.125f);
CGRect crispRect = CGRectIntegral(blurryRect);
```

11.4 Generating a QR Code

```
NSString *message = @"...";
CIFilter *QRCodeGenerator = [CIFilter filterWithName:@"CIQRCodeGenerator"];
[QRCodeGenerator setValue:[message dataUsingEncoding:NSUTF8StringEncoding]
                  forKey:@"inputMessage"];
[QRCodeGenerator setValue:@"H"
                  forKey:@"inputCorrectionLevel"];
```

11.5 Reading a QR Code

```
AVCaptureSession *session = [[AVCaptureSession alloc] init];
AVCaptureDevice *device =
    [AVCaptureDevice defaultDeviceWithMediaType:AVMediaTypeVideo];
NSError *error = nil;


AVCaptureDeviceInput *input =
    [AVCaptureDeviceInput deviceInputWithDevice:device error:&error];
if (input) {
    [session addInput:input];
} else {
    NSLog(@"Error: %@", error);
}

AVCaptureMetadataOutput *output =
    [[AVCaptureMetadataOutput alloc] init];
[session addOutput:output];
[output setMetadataObjectsDelegate:self queue:dispatch_get_main_queue()];
[output setMetadataObjectTypes:@[AVMetadataObjectTypeQRCode]];

[session startRunning];
```

```
#pragma mark - AVCaptureMetadataOutputObjectsDelegate

- (void)captureOutput:(AVCaptureOutput *)captureOutput
didOutputMetadataObjects:(NSArray *)metadataObjects
       fromConnection:(AVCaptureConnection *)connection
{
    NSString *QRCode = nil;
    for (AVMetadataObject *metadata in metadataObjects) {
        if ([metadata.type isEqualToString:AVMetadataObjectTypeQRCode]) {
            QRCode = [(AVMetadataMachineReadableCodeObject *)metadata ↩
   stringValue];
            break;
        }
    }

    NSLog(@"QR Code: %@", QRCode);
}
```

11.6 Creating an Image for a Swatch of Color

```
static UIImage * UIImageForSwatchOfColorWithSize(UIColor *color,
                                                 CGSize size)
{
    UIImage *image = nil;

    CGRect rect = CGRectMake(0.0f, 0.0f, size.width, size.height);

    UIGraphicsBeginImageContext(rect.size);
    {
        CGContextRef c = UIGraphicsGetCurrentContext();

        CGContextSetFillColorWithColor(c, [color CGColor]);
        CGContextFillRect(c, rect);

        image = UIGraphicsGetImageFromCurrentImageContext();
    }
    UIGraphicsEndImageContext();

    return image;
}
```

11.7 Cropping an Image to Face

```
@import CoreImage;

UIImage *image;
NSDictionary *options = @{CIDetectorAccuracy: CIDetectorAccuracyLow};
CIDetector *faceDetector = [CIDetector detectorOfType:CIDetectorTypeFace
                                              context:nil
                                              options:options];

CGRect bounds = CGRectZero;
NSArray *features =
    [faceDetector featuresInImage:
        [CIImage imageWithCGImage:[image CGImage]] options:nil];

for (CIFeature *feature in features) {
```

```
    CGRectUnion(bounds, feature.bounds);
}
```

11.8 Detecting a Face in an Image

```
NSDictionary *options = @{CIDetectorAccuracy: CIDetectorAccuracyLow};
CIDetector *smileDetector = [CIDetector detectorOfType:CIDetectorTypeFace
                                    context:context
                                    options:options];
NSArray *features =
    [smileDetector featuresInImage:image
                           options:@{CIDetectorSmile:@YES}];

if (([features count] > 0) &&
    (([(CIFaceFeature *)features[0]) hasSmile]))
{
    UIImageWriteToSavedPhotosAlbum(image,
                                   self,
                                   @selector(didFinishWritingImage),
                                   features);
} else {
    self.label.text = @"Say Cheese!"
}
```

11.9 Resizing an Image

UIGraphics

```
UIImage *image;
CGSize size;

UIImage *resizedImage;
UIGraphicsBeginImageContextWithOptions(size, NO, 0.0);
{
    [image drawInRect:CGRectMake(0.0f, 0.0f, size.width, size.height)];
    resizedImage = UIGraphicsGetImageFromCurrentImageContext();
}
UIGraphicsEndImageContext();
```

Image I/O

```
@import ImageIO;

CGImageSourceRef imageSource = CGImageSourceCreateWithData((__bridge ↩
   CFDataRef)data, NULL);
CGImageRef resizeImageRef = CGImageRef CGImageSourceCreateThumbnailAtIndex( ↩
   imageSource, 0, NULL);
```

11.10 Converting an Image to Data

PNG

```
UIImage *image = ...;
UIImagePNGRepresentation(image);
```

JPEG

```
UIImage *image = ...;
UIImageJPEGRepresentation(image, 0.7);
```

Quartz Bitmap

```
@import ImageIO;

CGDataProviderRef dataProvider = CGImageGetDataProvider(screenShot);
NSData *data = (__bridge_transfer NSData *)CGDataProviderCopyData( ↩
   dataProvider);
CGDataProviderRelease(dataProvider);
```

11.11 Determining the File Type of Image

Image Header

```
@import MobileCoreServices;

static CFStringRef UTTypeForImageData(NSData *data) {
    const unsigned char * bytes = [data bytes];

    if (data.length >= 8) {
```

```
        if (bytes[0] == 0x89 &&
            bytes[1] == 0x50 &&
            bytes[2] == 0x4E &&
            bytes[3] == 0x47 &&
            bytes[4] == 0x0D &&
            bytes[5] == 0x0A &&
            bytes[6] == 0x1A &&
            bytes[7] == 0x0A)
        {
            return kUTTypePNG;
        }
    }

    if (data.length >= 4) {
        if (bytes[0] == 0xFF &&
            bytes[1] == 0xD8 &&
            bytes[2] == 0xFF &&
            bytes[3] == 0xE0)
        {
            return kUTTypeJPEG;
        }
    }

    if (data.length >= 3) {
        if (bytes[0] == 0x47 &&
            bytes[1] == 0x49 &&
            bytes[2] == 0x46)
        {
            return kUTTypeGIF;
        } else if (bytes[0] == 0x49 &&
                   bytes[1] == 0x49 &&
                   bytes[2] == 0x2A)
        {
            return kUTTypeBMP;
        }
    }

    if (data.length >= 2) {
        if (bytes[0] == 0x42 &&
            bytes[1] == 0x4D)
        {
```

```
            return kUTTypeBMP;
        }
    }

    return nil;
}
```

Image I/O

```
@import ImageIO;

static CFStringRef UTTypeForImageData(NSData *data) {
    CFStringRef type = NULL;
    CGImageSourceRef imageSource = CGImageSourceCreateWithData((__bridge ↩
   CFDataRef)data, NULL);
    if (imageSource) {
        type = CGImageSourceGetType(imageSource);
        CFRelease(imageSource);
    }

    return type;
}
```

11.12 Applying a CIFilter on an Image

```
UIImage *originalImage = ...;

CIImage *inputImage = [[CIImage alloc] initWithImage:originalImage];
CIImage *outputImage = nil;

CGAffineTransform transform = CGAffineTransformIdentity;
CIFilter *clampFilter = [CIFilter filterWithName:@"CIAffineClamp"];
[clampFilter setDefaults];
[clampFilter setValue:[NSValue valueWithBytes:&transform objCType:@encode( ↩
   CGAffineTransform)] forKeyPath:@"inputTransform"];
[clampFilter setValue:inputImage forKeyPath:kCIInputImageKey];
outputImage = [clampFilter outputImage];

CGFloat darkness = 0.5f;
```

```
CIFilter *blackColorGenerator = [CIFilter filterWithName:@" ↩
   CIConstantColorGenerator"];
CIColor *blackColor = [[[UIColor blackColor] colorWithAlphaComponent: ↩
   darkness] CIColor];
[blackColorGenerator setValue:blackColor forKey:kCIInputColorKey];

CIFilter *compositeFilter = [CIFilter filterWithName:@"CIDarkenBlendMode"];
[compositeFilter setDefaults];
[compositeFilter setValue:[blackColorGenerator outputImage] forKey: ↩
   kCIInputImageKey];
[compositeFilter setValue:outputImage forKey:kCIInputBackgroundImageKey];
outputImage = [compositeFilter outputImage];

CGFloat radius = 2.0f;
CIFilter *blurFilter = [CIFilter filterWithName:@"CIGaussianBlur"];
[blurFilter setDefaults];
[blurFilter setValue:@(radius) forKey:kCIInputRadiusKey];
[blurFilter setValue:outputImage forKey:kCIInputImageKey];
outputImage = [blurFilter outputImage];

CIContext *context = [CIContext contextWithOptions:nil];
CGImageRef imageRef = [context createCGImage:outputImage fromRect:inputImage ↩
   .extent];
UIImage *filteredImage = [UIImage imageWithCGImage:imageRef];
CFRelease(imageRef);
```

11.13 Rounding Corners of a View

```
@import QuartzCore;

UIView *view = ...;
view.layer.cornerRadius = 5.0;
view.clipsToBounds = YES;
```

11.14 Getting Color RGB Components

```
UIColor *color = ...;
```

```
CGFloat r, g, b, a;
[color getRed:&r green:&g blue:&b alpha:&a];
```

11.15 Lightening / Darkening a Color

```
UIColor *color = ...;

CGFloat hue, saturation, brightness, alpha;
[color getHue:&hue
   saturation:&saturation
   brightness:&brightness
        alpha:&alpha];

CGFloat amount = 0.5;

UIColor *lighterColor =
    [UIColor colorWithHue:hue
               saturation:saturation * (1.0f - amount)
               brightness:brightness * (1.0f + amount)
                    alpha:alpha];

UIColor *darkerColor =
    [UIColor colorWithHue:hue
               saturation:saturation * (1.0f + amount)
               brightness:brightness * (1.0f - amount)
                    alpha:alpha];
```

11.16 Creating a Color from Hexadecimal String

```
NSScanner *scanner = [NSScanner scannerWithString:string];
scanner.charactersToBeSkipped =
    [[NSCharacterSet alphanumericCharacterSet] invertedSet];

unsigned value;
[scanner scanHexInt:&value];

CGFloat r = ((value & 0xFF0000) >> 16) / 255.0f;
CGFloat g = ((value & 0xFF00) >> 8) / 255.0f;
```

```
CGFloat b = ((value & 0xFF)) / 255.0f;

return [UIColor colorWithRed:r green:g blue:b alpha:1.0];
```

11.17 Inverting a UIColor

```
UIColor *color = ...;

CGFloat r, g, b, a;
[color getRed:&r green:&g blue:&b alpha:&a];

return [UIColor colorWithRed:1.0f - r
                       green:1.0f - g
                        blue:1.0f - b
                       alpha:a];
```

Chapter 12

Networking

12.1 Making a Request with CFNetwork

```
CFStringRef method = CFSTR("GET");

CFURLRef URL =
    CFURLCreateWithString(kCFAllocatorDefault,
                          CFSTR("http://nshipster.com/"),
                          NULL);

CFHTTPMessageRef request =
    CFHTTPMessageCreateRequest(kCFAllocatorDefault,
                               method,
                               URL,
                               kCFHTTPVersion1_1);

CFHTTPMessageSetHeaderFieldValue(request,
                                 CFSTR("Host"),
                                 CFAutorelease(CFURLCopyHostName(URL)));

CFHTTPMessageSetHeaderFieldValue(request,
                                 CFSTR("Accept"),
                                 CFSTR("text/html"));
CFHTTPMessageSetHeaderFieldValue(request,
                                 CFSTR("Content-Type"),
                                 CFSTR("charset=utf-8"));

CFReadStreamRef requestStream =
```

```
    CFReadStreamCreateForHTTPRequest(kCFAllocatorDefault, request);
CFMutableDataRef mutableResponseData =
    CFDataCreateMutable(kCFAllocatorDefault, 0);

CFIndex numberOfBytesRead = 0;
CFReadStreamOpen(requestStream);
do {
    uint8_t buffer[1024];
    numberOfBytesRead = CFReadStreamRead(requestStream,
                                         buffer,
                                         sizeof(buffer));
    if (numberOfBytesRead > 0) {
        CFDataAppendBytes(mutableResponseData, buffer, numberOfBytesRead);
    }
} while (numberOfBytesRead > 0);
CFReadStreamClose(requestStream);

CFHTTPMessageRef response =
    (CFHTTPMessageRef)CFReadStreamCopyProperty(requestStream,  ↩
   kCFStreamPropertyHTTPResponseHeader);
CFHTTPMessageSetBody(response, (CFDataRef)mutableResponseData);

CFIndex statusCode = CFHTTPMessageGetResponseStatusCode(response);
CFStringRef responseString =
    CFStringCreateWithBytes(kCFAllocatorDefault,
                            CFDataGetBytePtr(mutableResponseData),
                            CFDataGetLength(mutableResponseData),
                            kCFStringEncodingUTF8,
                            NO);

CFRelease(requestStream);
CFRelease(mutableResponseData);
CFRelease(response);
CFRelease(request);
CFRelease(URL);
```

12.2 Creating a Bound NSStream Pair

```
CFReadStreamRef readStream = NULL;
```

```
CFWriteStreamRef writeStream = NULL;

CFIndex bufferSize = 4096;

CFStreamCreateBoundPair(NULL, &readStream, &writeStream, bufferSize);

NSInputStream *inputStream = CFBridgingRelease(readStream);
NSOutputStream *outputStream = CFBridgingRelease(writeStream);
```

12.3 Constructing an HTTP User-Agent

iOS

```
NSString *applicationName =
  [[[NSBundle mainBundle] infoDictionary]
      objectForKey:(__bridge NSString *)kCFBundleExecutableKey];

NSString *applicationVersion =
  [[[NSBundle mainBundle] infoDictionary]
      objectForKey:(__bridge NSString *)kCFBundleVersionKey];

NSString *deviceModel = [[UIDevice currentDevice] model];
NSString *systemVersion = [[UIDevice currentDevice] systemVersion];
NSString *screenScale = [[UIScreen mainScreen] scale]];

[NSString stringWithFormat:@"%@/%@ (%@; iOS %@; Scale/%0.2f)", ↩
   applicationName, applicationVersion, deviceModel, systemVersion, ↩
   screenScale];
```

Mac OS X

```
NSString *applicationName =
  [[[NSBundle mainBundle] infoDictionary]
    objectForKey:(__bridge NSString *)kCFBundleExecutableKey];

NSString *applicationVersion =
  [[[NSBundle mainBundle] infoDictionary]
    objectForKey:(__bridge NSString *)kCFBundleIdentifierKey];

NSString *systemVersion =
  [[[NSBundle mainBundle] infoDictionary]
```

```
    objectForKey:@"CFBundleShortVersionString"]

[NSString stringWithFormat:@"%@/%@ (%@; iOS %@; Scale/%0.2f)",  ↩
   applicationName, applicationVersion, systemVersion];
```

12.4 Determining the MIME Type for a File Extension

```
@import MobileCoreServices;

NSString *path = @"/path/to/document.json";
NSString *UTI =
    (__bridge_transfer NSString *)
      UTTypeCreatePreferredIdentifierForTag(
                                  kUTTagClassFilenameExtension,
                                  (__bridge CFStringRef)[path pathExtension ↩
   ],
                                  NULL);
NSString *MIMEType =
  (__bridge_transfer NSString *)
      UTTypeCopyPreferredTagWithClass(
                                  (__bridge CFStringRef)UTI,
                                  kUTTagClassMIMEType);
```

12.5 Setting HTTP Headers for a URL Request

```
NSURL *URL = [NSURL URLWithString:@"http://example.com"];
NSMutableURLRequest *mutableRequest = [NSMutableURLRequest requestWithURL: ↩
   URL];
[mutableRequest setValue:@"application/json"
      forHTTPHeaderField:@"Accept"];
```

12.6 Escaping URLs

```
NSString *string = ...;
NSString *URLEscapedString =
  [string stringByAddingPercentEncodingWithAllowedCharacters:
```

```
    [NSCharacterSet URLQueryAllowedCharacterSet]];
```

12.7 Building a URL relative to a Base URL

```
NSURL *baseURL = [NSURL URLWithString:@"http://example.com/v1/"];

// http://example.com/v1/foo
[NSURL URLWithString:@"foo"
       relativeToURL:baseURL];

// http://example.com/v1/foo?bar=baz
[NSURL URLWithString:@"foo?bar=baz"
       relativeToURL:baseURL];

// http://example.com/foo
[NSURL URLWithString:@"/foo"
       relativeToURL:baseURL];

// http://example.com/v1/foo
[NSURL URLWithString:@"foo/"
       relativeToURL:baseURL];

// http://example.com/foo/
[NSURL URLWithString:@"/foo/"
       relativeToURL:baseURL];

// http://example2.com/
[NSURL URLWithString:@"http://example2.com/"
       relativeToURL:baseURL];
```

12.8 Setting the Shared URL Cache

```
NSURLCache *URLCache =
  [[NSURLCache alloc] initWithMemoryCapacity:4 * 1024 * 1024
                                diskCapacity:20 * 1024 * 1024
                                    diskPath:nil];
[NSURLCache setSharedURLCache:URLCache];
```

12.9 Making Asynchronous Network Request with NSURLConnection

```
NSURL *URL = [NSURL URLWithString:@"http://example.com"];
NSURLRequest *request = [NSURLRequest requestWithURL:URL];
[NSURLConnection sendAsynchronousRequest:request
                                   queue:[NSOperationQueue mainQueue]
                       completionHandler:
^(NSURLResponse *response, NSData *data, NSError *connectionError) {
    // ...
}];
```

12.10 Making Synchronous Network Request with NSURLConnection

DON'T. All networking should be done asynchronously. === Handling Authentication Challenges ===

```
#pragma mark - NSURLConnectionDelegate

- (void)connection:(NSURLConnection *)connection
willSendRequestForAuthenticationChallenge:(NSURLAuthenticationChallenge *) ↩
   challenge
{
    if ([challenge.protectionSpace.authenticationMethod isEqualToString: ↩
   NSURLAuthenticationMethodServerTrust]) {
        SecTrustResultType result;
        OSStatus status =
            SecTrustEvaluate(challenge.protectionSpace.serverTrust,
                             &result);
        BOOL isTrustValid = status == noErr &&
          (result == kSecTrustResultUnspecified ||
           result == kSecTrustResultProceed);

        if (isTrustValid) {
            NSURLCredential *credential =
              [NSURLCredential
                  credentialForTrust:challenge.protectionSpace.serverTrust];
            [[challenge sender] useCredential:credential
```

```
                    forAuthenticationChallenge:challenge];
        } else {
            [[challenge sender] cancelAuthenticationChallenge:challenge];
        }
    } else {
        if ([challenge previousFailureCount] == 0) {
            if (self.credential) {
                [[challenge sender] useCredential:self.credential
                        forAuthenticationChallenge:challenge];
            } else {
                [[challenge sender] ↩
   continueWithoutCredentialForAuthenticationChallenge:challenge];
            }
        } else {
            [[challenge sender] ↩
   continueWithoutCredentialForAuthenticationChallenge:challenge];
        }
    }
}
```

12.11 Making Asynchronous Network Request with NSURLSession

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

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

[task resume];
```

12.12 Getting List of Network Interfaces

```
@import SystemConfiguration;
```

```
SCDynamicStoreContext context = { 0, NULL, NULL, NULL, NULL };
SCDynamicStoreRef store = SCDynamicStoreCreate(NULL, NULL, nil, &context);
CFPropertyListRef propertyList = SCDynamicStoreCopyValue(store, CFSTR("State ↩
   :/Network/Interface"));
NSArray *interfaces = (__bridge NSArray *)CFDictionaryGetValue(propertyList, ↩
    CFSTR("Interfaces"));
```

Example Output

```
- lo0
- gif0
- stf0
- en0
- en1
- en2
- bridge0
- p2p0
- utun0
```

12.13 Monitoring Network Reachability

```
static void ReachabilityCallback(SCNetworkReachabilityRef target,  ↩
   SCNetworkConnectionFlags flags, void* info) {
   // ...
}

BOOL ignoresAdHocWiFi = NO;

struct sockaddr_in ipAddress;
bzero(&ipAddress, sizeof(ipAddress));
ipAddress.sin_len = sizeof(ipAddress);
ipAddress.sin_family = AF_INET;
ipAddress.sin_addr.s_addr =
  htonl(ignoresAdHocWiFi ? INADDR_ANY : IN_LINKLOCALNETNUM);

SCNetworkReachabilityRef reachability =
    SCNetworkReachabilityCreateWithAddress(kCFAllocatorDefault,
                                           (struct sockaddr *)&ipAddress);

SCNetworkReachabilityContext context = {0, NULL, NULL, NULL, NULL};
```

```
if (SCNetworkReachabilitySetCallback(reachability,
                                     ReachabilityCallback,
                                     &context))
{
   if (!SCNetworkReachabilityScheduleWithRunLoop(reachability,
                                                 CFRunLoopGetMain(),
                                                 kCFRunLoopCommonModes))
{
      SCNetworkReachabilitySetCallback(reachability, NULL, NULL);
      return nil;
   }
}

SCNetworkConnectionFlags flags;
if (SCNetworkReachabilityGetFlags(reachability, &flags)) {
  BOOL isReachable =
    ((flags & kSCNetworkReachabilityFlagsReachable) != 0);

  BOOL needsConnection =
    ((flags & kSCNetworkReachabilityFlagsConnectionRequired) != 0);

  BOOL canConnectionAutomatically =
    (((flags & kSCNetworkReachabilityFlagsConnectionOnDemand ) != 0) ||
      ((flags & kSCNetworkReachabilityFlagsConnectionOnTraffic) != 0));

  BOOL canConnectWithoutUserInteraction =
    (canConnectionAutomatically &&
      (flags & kSCNetworkReachabilityFlagsInterventionRequired) == 0);

  BOOL isNetworkReachable =
    (isReachable &&
      (!needsConnection || canConnectWithoutUserInteraction));
}
```

12.14 Validating an SSL Certificate

```
SecTrustRef trust;
SecPolicyRef X509Policy = SecPolicyCreateBasicX509();
```

```
SecTrustSetPolicies(serverTrust, (__bridge CFArrayRef)@[(__bridge id) ↩
   X509Policy]);

SecTrustResultType result;
__Require_noErr(SecTrustEvaluate(trust, &result), exit);
```

12.15 Adding a URL to the Safari Reading List

```
@import SafariServices;

NSURL *URL = [NSURL URLWithString:@"http://nshipster.com/"];
[[SSReadingList defaultReadingList] addReadingListItemWithURL:URL
                                                        title:@"NSHipster"
                                                  previewText:@"..."
                                                        error:nil];
```

Chapter 13

UIKit

13.1 Determining the Current Device Model

```
#import <sys/sysctl.h>

static NSString * AAPLCurrentDeviceModelIdentifier() {
    static char * const AAPLHardwareModelTypeSpecifier = "hw.machine";

    size_t size;
    [source,Objective-C 2.0](AAPLHardwareModelTypeSpecifier, NULL, &size, ↩
   NULL, 0);

    char *result = malloc(size);
    [source,Objective-C 2.0](AAPLHardwareModelTypeSpecifier, result, &size, ↩
   NULL, 0);

    NSString *identifier = [NSString stringWithCString:result encoding: ↩
   NSUTF8StringEncoding];
    free(result);

    return identifier;
}

static NSString * AAPLModelNameForIdentifier(NSString *identifier) {
    if ([identifier hasPrefix:@"iPhone"]) {
        if ([identifier isEqualToString:@"iPhone1,1"]) {
            return @"iPhone 1G";
        } else if ([identifier isEqualToString:@"iPhone1,2"]) {
```

```
            return @"iPhone 3G";
        } else if ([identifier isEqualToString:@"iPhone2,1"]) {
            return @"iPhone 3GS";
        } else if ([identifier isEqualToString:@"iPhone3,1"]) {
            return @"iPhone 4 (GSM)";
        } else if ([identifier isEqualToString:@"iPhone3,2"]) {
            return @"iPhone 4 (GSM Rev A)";
        } else if ([identifier isEqualToString:@"iPhone3,3"]) {
            return @"iPhone 4 (CDMA)";
        } else if ([identifier isEqualToString:@"iPhone4,1"]) {
            return @"iPhone 4S";
        } else if ([identifier isEqualToString:@"iPhone5,1"]) {
            return @"iPhone 5 (GSM)";
        } else if ([identifier isEqualToString:@"iPhone5,2"]) {
            return @"iPhone 5 (Global)";
        } else if ([identifier isEqualToString:@"iPhone5,3"]) {
            return @"iPhone 5C (GSM)";
        } else if ([identifier isEqualToString:@"iPhone5,4"]) {
            return @"iPhone 5C (Global)";
        } else if ([identifier isEqualToString:@"iPhone6,1"]) {
            return @"iPhone 5S (GSM)";
        } else if ([identifier isEqualToString:@"iPhone6,2"]) {
            return @"iPhone 5S (Global)";
        }
    }

    if ([identifier hasPrefix:@"iPod"]) {
        if ([identifier isEqualToString:@"iPod1,1"]) {
            return @"iPod Touch 1G";
        } else if ([identifier isEqualToString:@"iPod2,1"]) {
            return @"iPod Touch 2G";
        } else if ([identifier isEqualToString:@"iPod3,1"]) {
            return @"iPod Touch 3G";
        } else if ([identifier isEqualToString:@"iPod4,1"]) {
            return @"iPod Touch 4G";
        } else if ([identifier isEqualToString:@"iPod5,1"]) {
            return @"iPod Touch 5G";
        }
    }

    if ([identifier hasPrefix:@"iPad"]) {
```

```
    if ([identifier isEqualToString:@"iPad1,1"]) {
        return @"iPad 1G";
    } else if ([identifier isEqualToString:@"iPad2,1"]) {
        return @"iPad 2 (WiFi)";
    } else if ([identifier isEqualToString:@"iPad2,2"]) {
        return @"iPad 2 (GSM)";
    } else if ([identifier isEqualToString:@"iPad2,3"]) {
        return @"iPad 2 (CDMA)";
    } else if ([identifier isEqualToString:@"iPad2,4"]) {
        return @"iPad 2 (Rev A)";
    } else if ([identifier isEqualToString:@"iPad3,1"]) {
        return @"iPad 3 (WiFi)";
    } else if ([identifier isEqualToString:@"iPad3,2"]) {
        return @"iPad 3 (GSM)";
    } else if ([identifier isEqualToString:@"iPad3,3"]) {
        return @"iPad 3 (Global)";
    } else if ([identifier isEqualToString:@"iPad3,4"]) {
        return @"iPad 4 (WiFi)";
    } else if ([identifier isEqualToString:@"iPad3,5"]) {
        return @"iPad 4 (GSM)";
    } else if ([identifier isEqualToString:@"iPad3,6"]) {
        return @"iPad 4 (Global)";
    } else if ([identifier isEqualToString:@"iPad4,1"]) {
        return @"iPad Air (WiFi)";
    } else if ([identifier isEqualToString:@"iPad4,2"]) {
        return @"iPad Air (Cellular)";
    }

    if ([identifier isEqualToString:@"iPad2,5"]) {
        return @"iPad Mini 1G (WiFi)";
    } else if ([identifier isEqualToString:@"iPad2,6"]) {
        return @"iPad Mini 1G (GSM)";
    } else if ([identifier isEqualToString:@"iPad2,7"]) {
        return @"iPad Mini 1G (Global)";
    } else if ([identifier isEqualToString:@"iPad4,4"]) {
        return @"iPad Mini Retina (WiFi)";
    } else if ([identifier isEqualToString:@"iPad4,5"]) {
        return @"iPad Mini Retina (Cellular)";
    }
}
```

```
    if ([identifier isEqualToString:@"x86_64"] || [identifier hasSuffix:@"86 ↩
   "]) {
        BOOL hasWideScreen = ([[UIScreen mainScreen] bounds].size.width >=  ↩
   768.0);
        return (hasWideScreen ? @"iPad Simulator" : @"iPhone Simulator");
    }

    return nil;
}
```

13.2 Forcing Screen Orientation

```
[[NSNotificationCenter defaultCenter]
    postNotificationName:UIDeviceOrientationDidChangeNotification
                  object:nil];
```

13.3 Making a Device Vibrate

```
@import AudioToolbox;

// Plays an alert noise if vibration not supported on device
AudioServicesPlayAlertSound(kSystemSoundID_Vibrate);

// No-op on devices that do not support vibration
AudioServicesPlaySystemSound(kSystemSoundID_Vibrate);
```

13.4 Implementing UITableViewDataSource

```
#pragma mark - UITableViewDataSource

- (NSInteger)numberOfSectionsInTableView:(UITableView *)tableView {
    return <#number#>;
}

- (NSInteger)tableView:(UITableView *)tableView
 numberOfRowsInSection:(NSInteger)section
```

```
{
    return <#number#>
}

- (UITableViewCell *)tableView:(UITableView *)tableView
         cellForRowAtIndexPath:(NSIndexPath *)indexPath
{
    UITableViewCell *cell = [tableView dequeueReusableCellWithIdentifier:<# ↩
   reuseIdentifier#> forIndexPath:<#indexPath#>];

    [self configureCell:cell forRowAtIndexPath:indexPath];

    return cell;
}

- (void)configureCell:(UITableViewCell *)cell
    forRowAtIndexPath:(NSIndexPath *)indexPath
{
    <#statements#>
}
```

13.5 Implementing UITableViewDelegate

```
#pragma mark - UITableViewDelegate

- (void)tableView:(UITableView *)tableView
didSelectRowAtIndexPath:(NSIndexPath *)indexPath
{
    <#statements#>
}
```

13.6 Using iOS 6 Styles for Standard Controls in iOS 7 App

```
[[NSUserDefaults standardUserDefaults] setObject:@YES
                                          forKey:@"UIUseLegacyUI"]
```

13.7 Creating a Snapshot of a View

```
UIView *view = ...;
double scale = [[UIScreen mainScreen] scale];

UIImage *snapshot = nil;
UIGraphicsBeginImageContextWithOptions(view.bounds.size, NO, scale);
{
  if ([self respondsToSelector:@selector(drawViewHierarchyInRect: ↩
   afterScreenUpdates:)]) {
    [self drawViewHierarchyInRect:view.bounds afterScreenUpdates:YES];
  } else {
    [self.layer renderInContext:UIGraphicsGetCurrentContext()];
  }

  snapshot = UIGraphicsGetImageFromCurrentImageContext();
}
UIGraphicsEndImageContext();
```

13.8 Determining if UIViewController is Visible

```
- (BOOL)isVisible {
    return [self isViewLoaded] && self.view.window;
}
```

13.9 Removing Bounce from UIWebView

```
webView.scrollView.bounces = NO;
```

13.10 Removing Drop Shadow from UIWebView

```
for (UIView *view in [[[webView subviews] objectAtIndex:0] subviews]) {
  if ([view isKindOfClass:[UIImageView class]] && view.frame.size.width ==  ↩
   1.0f) {
    view.hidden = YES;
```

```
    }
}
```

13.11 Preventing Links from Being Tapped in UIWebView

```
#pragma mark - UIWebViewDelegate

- (BOOL)webView:(UIWebView *)webView
shouldStartLoadWithRequest:(NSURLRequest *)request
navigationType:(UIWebViewNavigationType)navigationType
{
    return NO;
}
```