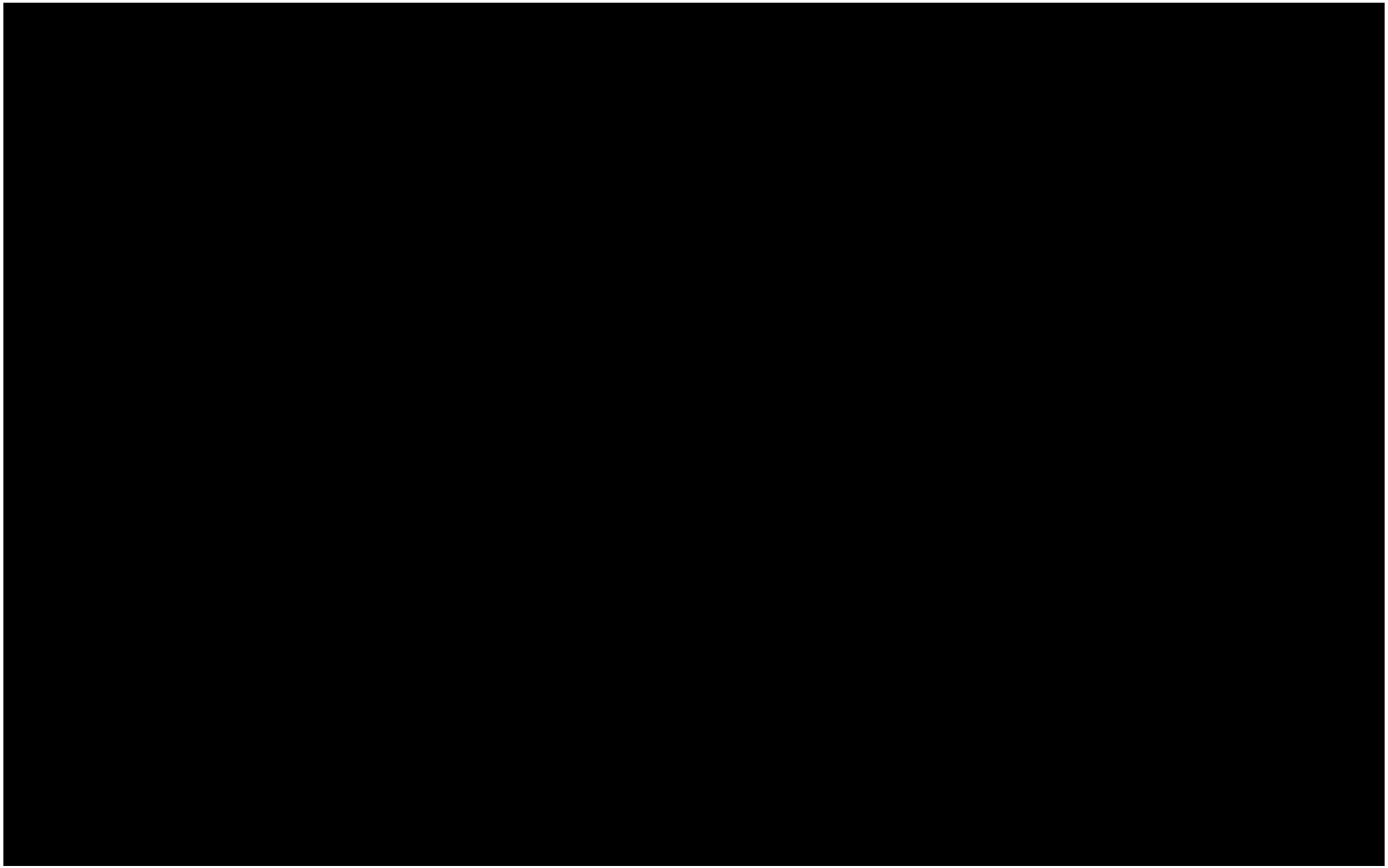
Migrating to Modern Objective-C

Session 413

Malcolm Crawford
Developer Publications

Rip van NSWinkle

Asleep since 2005



Where did all the money go?

Perhaps I should have used
that maid service.

What are all those little black
things everyone's tapping on?

Where can I get a haircut?

Where did all the money go?

Perhaps I should have used
Interface Builder.

What happened to Objective-C?

What are all those little black
things everyone's tapping on?

Where can I get a haircut?

Where did all the money go?

Perhaps I should have used
a personal service.

What happened to Objective-C?

What are all those little black
things everyone's tapping on?

Where can I get a haircut?

Where did all the money go?

Perhaps I should have used
a personal service.

What happened to Objective-C?

What are all those little black
things everyone's tapping on?

Where can I get a haircut?

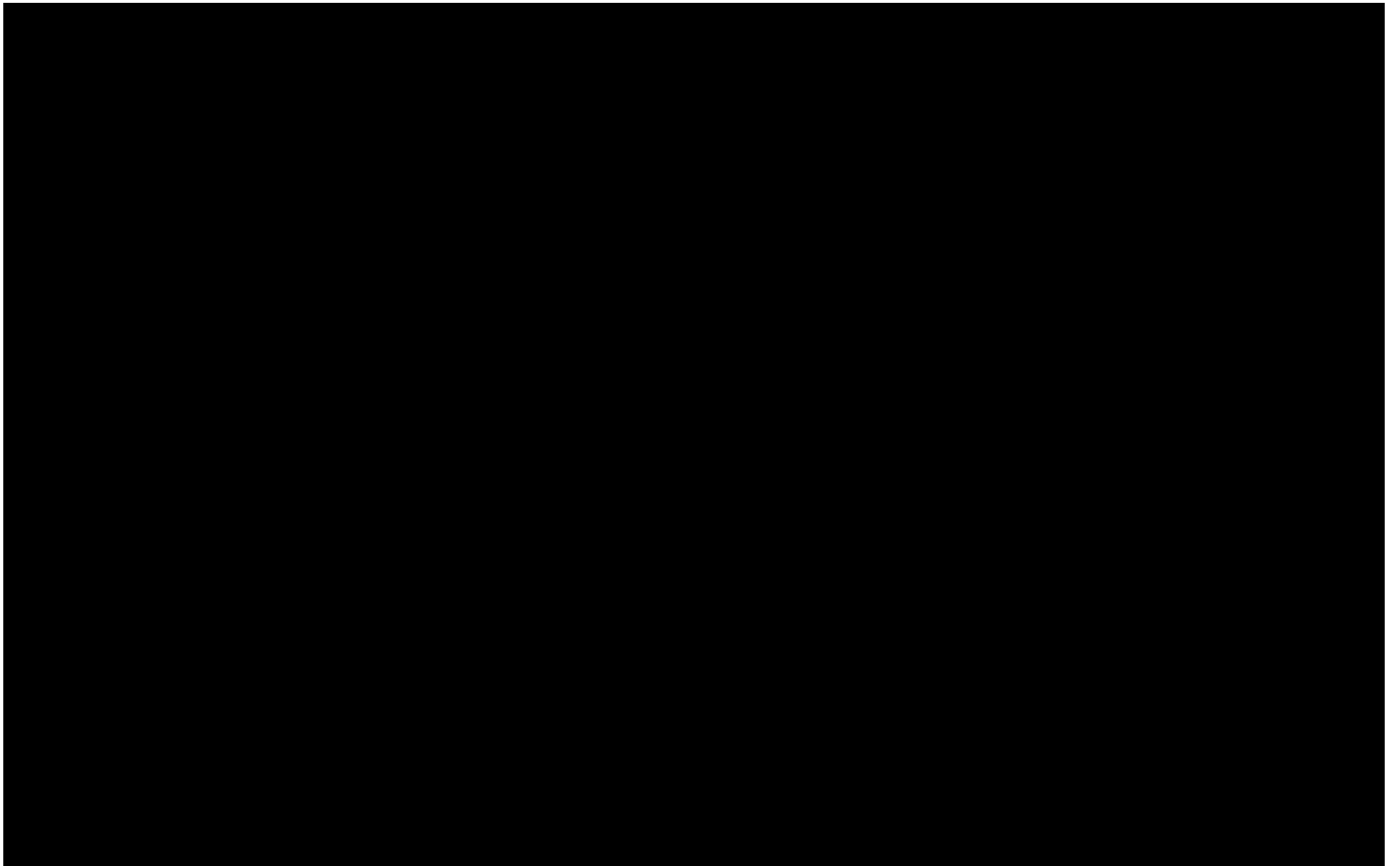
How do I use it?

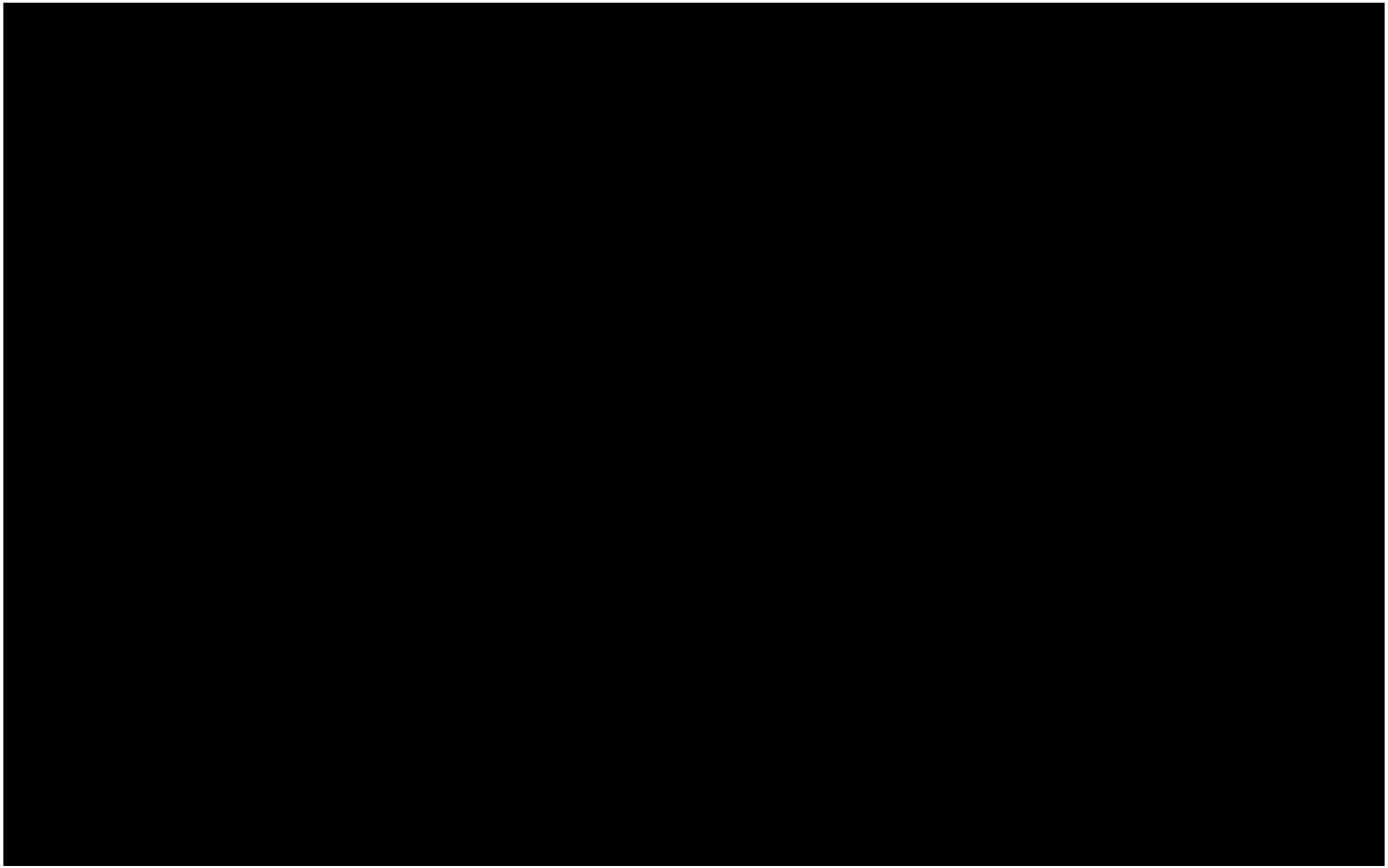
Agenda

- Philosophy
- Headers
- Enums
- Accessor methods
- Memory management
- Properties
- Initializers and dealloc
- Protocols
- Collections and literals
- Blocks

Objective-C Has Evolved

- Historically: Objective-C was a thin layer atop C
- Today: Objective-C is still atop C
  - But it's not such a thin layer

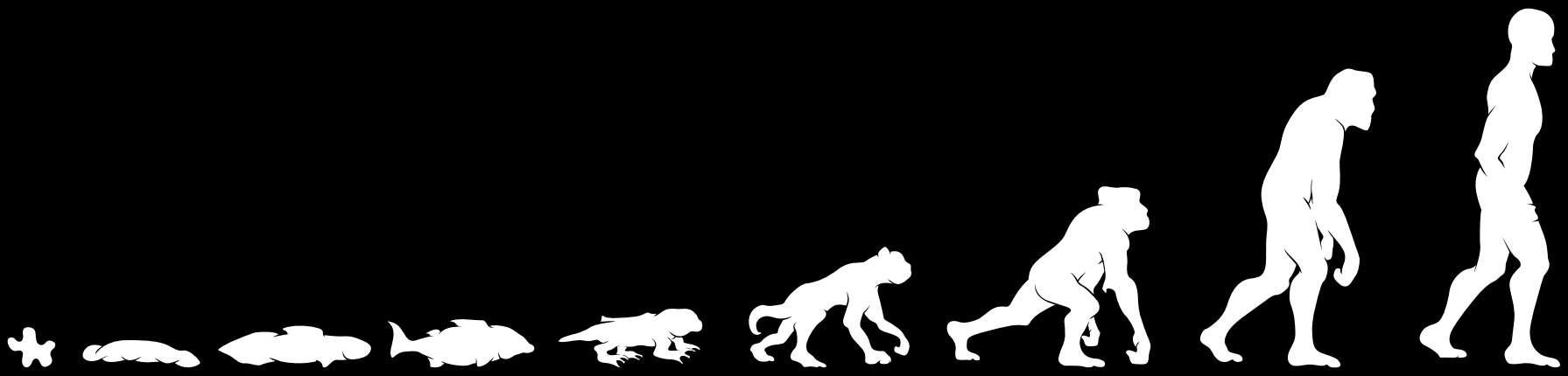




Evolution of Objective-C

Simpler, safer through automation

- Object Oriented C
- Retain counting
- Properties
- Blocks
- ARC



New (and Better) Practices

- With new language features have come new patterns
  - Conventions have evolved
  - Important—Cocoa has strong conventions
- New features make code more compact, expressive, correct, efficient
  - Remove unnecessary boilerplate
  - Clarify API contracts
  - Separate private from public
  - Build conventions into the language
    - So you don't have to think about them

“Civilization advances by extending the number of important operations which we can perform without thinking about them.”

Alfred North Whitehead, 1861-1947

Importing Headers

```
#import <Foundation/NSObject.h>
#import <Foundation/NSString.h>
//or @class NSString;
```

```
#import <Foundation/Foundation.h>
```

Enums with Fixed Underlying Type

```
enum {
  NSOperationQueuePriorityVeryLow = -8,
  NSOperationQueuePriorityLow = -4,
  NSOperationQueuePriorityNormal = 0,
  NSOperationQueuePriorityHigh = 4,
  NSOperationQueuePriorityVeryHigh = 8
};
```

```
enum {
   NSOperationQueuePriorityVeryLow = -8,
   NSOperationQueuePriorityLow = -4,
   NSOperationQueuePriorityNormal = 0,
   NSOperationQueuePriorityHigh = 4,
   NSOperationQueuePriorityVeryHigh = 8
};

typedef NSInteger NSOperationQueuePriority;
```

```
typedef enum : NSInteger {
  NSOperationQueuePriorityVeryLow = -8,
  NSOperationQueuePriorityLow = -4,
  NSOperationQueuePriorityNormal = 0,
  NSOperationQueuePriorityHigh = 4,
  NSOperationQueuePriorityVeryHigh = 8
} NSOperationQueuePriority;
```

Accessor Methods

```
- (void)myMethod {
    // ...
    [title release];
    title = [NSString stringWithFormat:@"Area: %1.2f", [self area]];
    [title retain];
    // ...
}
```

```
- (void)myMethod {
    // ...

    [self setTitle:[NSString stringWithFormat:@"Area: %1.2f", [self area]]];

    // ...
}
```

```
- (void)myMethod {
    // ...

    self.title = [NSString stringWithFormat:@"Area: %1.2f", self.area];

    // ...
}
```

Always use accessor methods

Always use accessor methods

Except in initializer methods and dealloc

Memory Management

```objc
NSMutableArray *array = [[NSMutableArray alloc] init];

// Use the array

[array release];
```

```objc
NSMutableArray *array = [[[NSMutableArray alloc] init] autoelease];

// Use the array
```

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

// Use the array
```

```
id heisenObject = [array objectAtIndex:n];
[array removeObjectAtIndex:n];
// ...
[heisenObject doSomething];
// ...
```

```
id heisenObject = [[array objectAtIndex:n] retain];
[array removeObjectAtIndex:n];
// ...
[heisenObject doSomething];
[heisenObject release];
// ...
```

```objc
// Person.m

- (void)takeLastNameFrom:(Person *)person
{
    NSString *oldLastname = [self lastName];
    [self setLastName:[person lastName]];
    NSLog(@"Lastname changed from %@ to %@", oldLastname, [self lastName]);
}
```

```
// Person.m

- (void)takeLastNameFrom:(Person *)person
{
    NSString *oldLastname = [[self lastName] retain];
    [self setLastName:[person lastName]];
    NSLog(@"Lastname changed from %@ to %@", oldLastname, [self lastName]);
    [oldLastName release];
}
```

Use ARC

Adopting Automatic Reference Counting	Marina Wednesday 11:30AM
Adopting Automatic Reference Counting (Repeat)	Nob Hill Friday 11:30AM

```
NSMutableArray *array = [[NSMutableArray alloc] init];

// Use the array
```

```
id heisenObject = [array objectAtIndex:n];
[array removeObjectAtIndex:n];
// ...
[heisenObject doSomething];
// ...
```

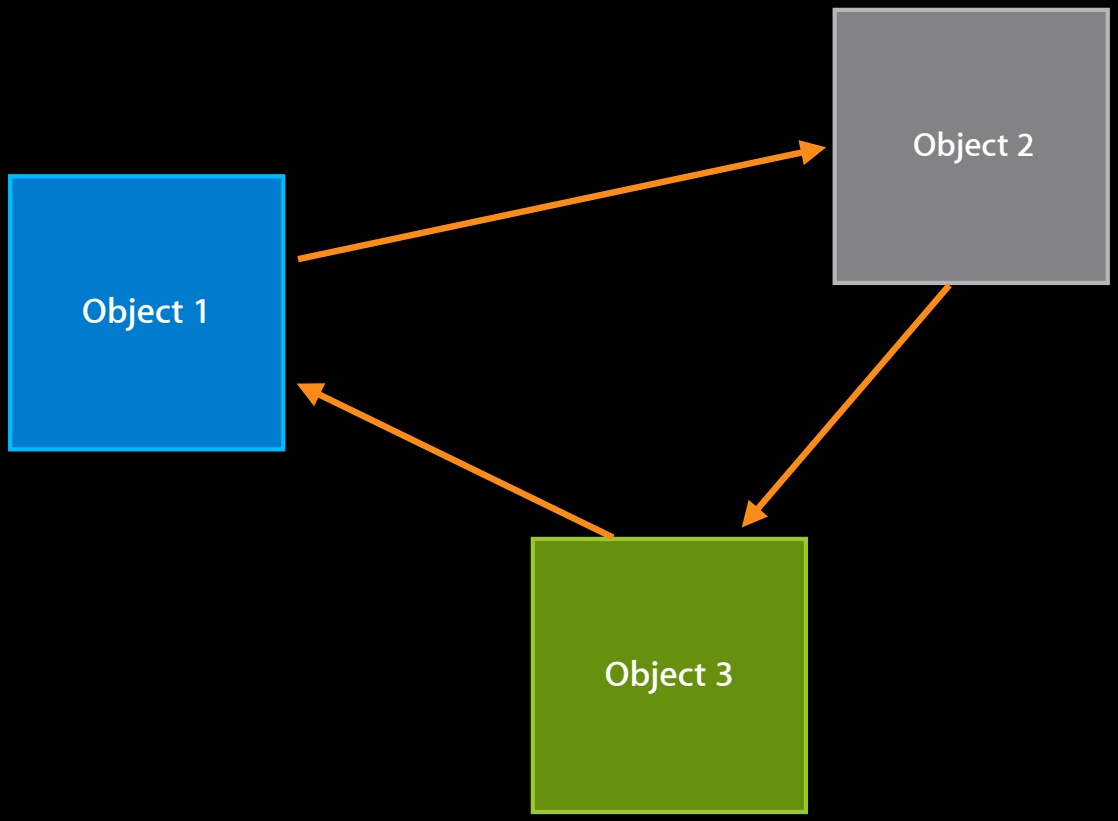
```
// Person.m

- (void)takeLastNameFrom:(Person *)person
{
    NSString *oldLastname = [self lastName];
    [self setLastName:[person lastName]];
    NSLog(@"Lastname changed from %@ to %@", oldLastname, [self lastName]);
}
```

```
CFUUIDRef cfUUID = CFUUIDCreate(NULL);
NSString *noteUUID = (NSString *)
                      CFUUIDCreateString(NULL, cfUUID);
CFRelease(cfUUID);
```

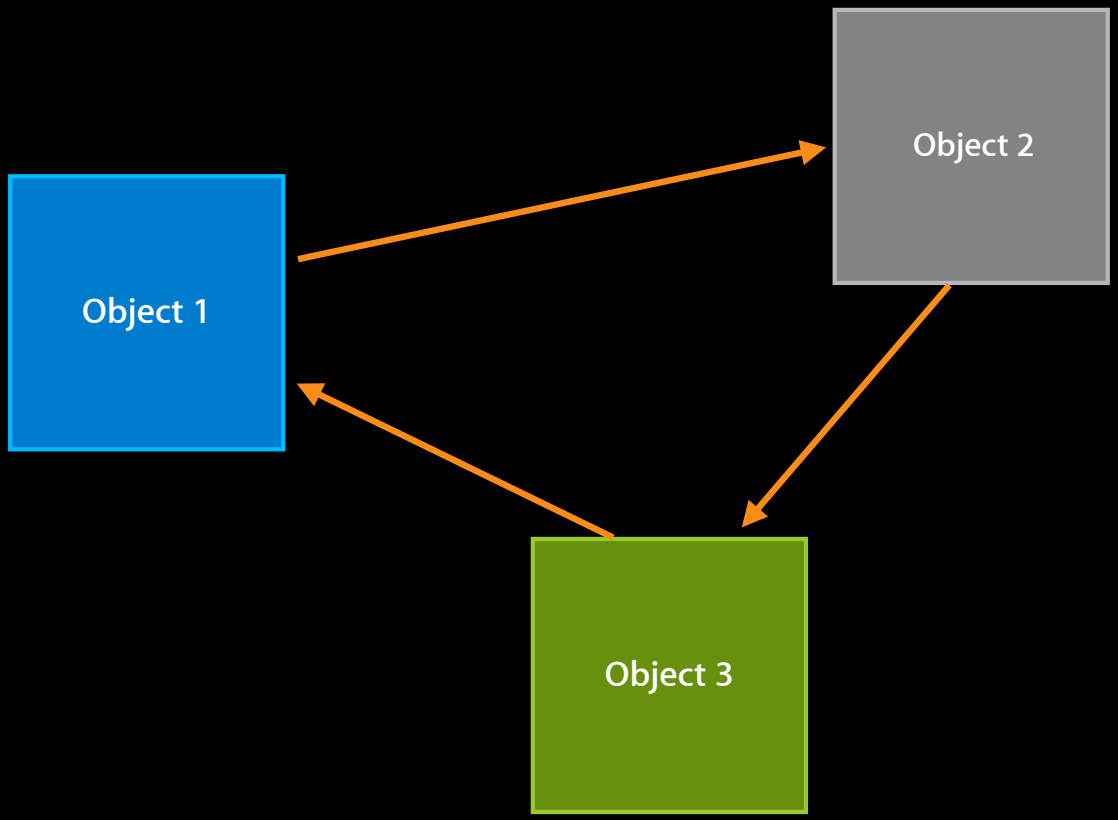
```
CFUUIDRef cfUUID = CFUUIDCreate(NULL);
NSString *noteUUID = (__bridge_transfer NSString *)
                      CFUUIDCreateString(NULL, cfUUID);
CFRelease(cfUUID);
```

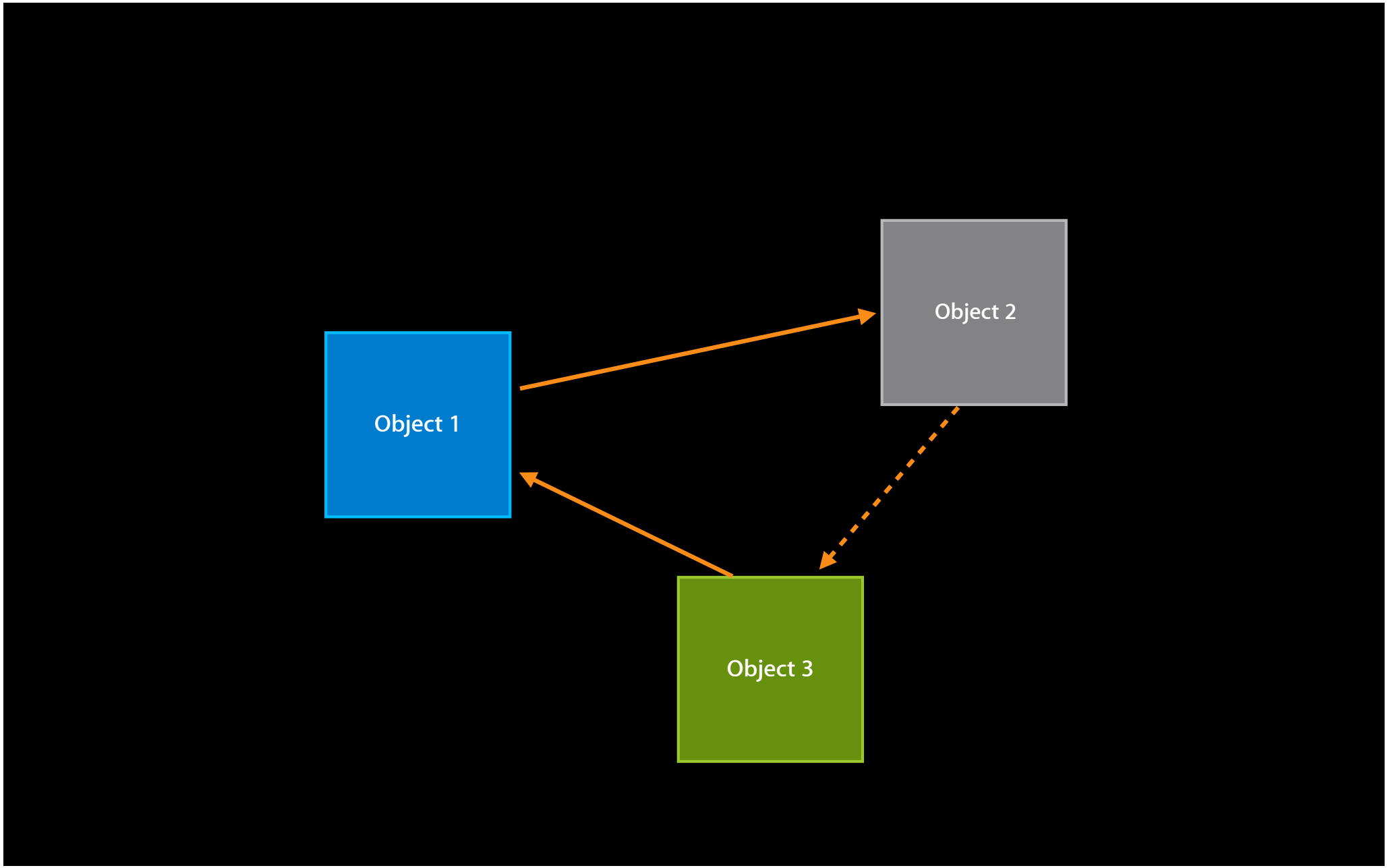
```
NSString *noteUUID = [[[NSUUID alloc] init] UUIDString];
```

Retain Cycle



Strong Reference Cycle




Object 1
Object 2
Object 3

Properties

The Old

```
@interface Thing : NSObject {
    NSString *title;
    float radius;
    id delegate;
}
- (float)area;
- (NSString *)title;
- (void)setTitle:(NSString *)newTitle;
- (float)radius;
- (void)setRadius:(float)newValue;
- (id)delegate;
- (void)setDelegate:(id)newDelegate;
@end
```

```
@implementation Thing
- (float)area {
    return M_PI * pow(radius, 2.0);
}
- (NSString *)title {
    return title;
}
- (void)setTitle:(NSString *)newTitle {
    if (title != newTitle) {
        [title release];
        title = [newTitle copy];
    }
}
// ....
```

The New

```
@interface Thing : NSObject
@property (copy) NSString *title;
@property float radius;
@property (weak) id delegate;
@property(readonly, nonatomic) float area;
@end
```

```
@implementation Thing
- (float)area {
    return M_PI * pow(self.radius, 2.0);
}
@end
```

```
@interface Thing : NSObject
{
   NSString *title;
   float radius;
   id delegate;
}
```

```
@interface Thing : NSObject
{
    NSString *title;
    float radius;
    id delegate;
}
```

```
@implementation Thing
{
    NSString *title;
    float radius;
    id delegate;
}
```

```
@interface Thing : NSObject
```

```
- (NSString *)title;
- (void)setTitle:(NSString *)newTitle;

- (float)radius;
- (void)setRadius:(float)newValue;
- (float)area;

- (id)delegate;
- (void)setDelegate:(id)newDelegate;
```

```
- (NSString *)title;
- (void)setTitle:(NSString *)newTitle;

- (float)radius;
- (void)setRadius:(float)newValue;
- (float)area;

- (id)delegate;
- (void)setDelegate:(id)newDelegate;
```

```
- (NSString *)title;
- (void)setTitle:(NSString *)newTitle;
```

```
@property NSString *title;
```

```
@property (copy) NSString *title;
```

```
@property NSString *title;


- (float)radius;
- (void)setRadius:(float)newValue;
- (float)area;

- (id)delegate;
- (void)setDelegate:(id)newDelegate;
```

```
@property NSString *title;


- (float)radius;
- (void)setRadius:(float)newValue;
- (float)area;

- (id)delegate;
- (void)setDelegate:(id)newDelegate;
```

```
- (float)radius;
- (void)setRadius:(float)newValue;
- (float)area;
```

```
@property float radius;

- (float)area;
```

```
@property float radius;

@property (readonly) float area;
```

```
@property float radius;

@property (readonly, nonatomic) float area;
```

```
@property NSString *title;
@property float radius;
@property (readonly, nonatomic) float area;



- (id)delegate;
- (void)setDelegate:(id)newDelegate;
```

```
@property NSString *title;
@property float radius;
@property (readonly, nonatomic) float area;
```

```
- (id)delegate;
- (void)setDelegate:(id)newDelegate;
```

```
- (id)delegate;
- (void)setDelegate:(id)newDelegate;
```

```
@property (weak) id delegate;
```

```
@interface Thing : NSObject {
    NSString *title;
    float radius;
    id delegate;
}
- (float)area;
- (NSString *)title;
- (void)setTitle:(NSString *)newTitle;
- (float)radius;
- (void)setRadius:(float)newValue;
- (id)delegate;
- (void)setDelegate:(id)newDelegate;
@end
```

```
@implementation Thing
- (float)area {
    return M_PI * pow(self.radius, 2.0);
}
- (NSString *)title {
    return title;
}
- (void)setTitle:(NSString *)newTitle {
    if (title != newTitle) {
        [title release];
        title = [newTitle copy];
    }
}
// ....
```

```objc
@interface Thing : NSObject {
    NSString *title;
    float radius;
    id delegate;
}
- (float)area;
- (NSString *)title;
- (void)setTitle:(NSString *)newTitle;
- (float)radius;
- (void)setRadius:(float)newValue;
- (id)delegate;
- (void)setDelegate:(id)newDelegate;
@end
```

```objc
@implementation Thing
- (float)area {
    return M_PI * pow(self.radius, 2.0);
}
- (NSString *)title {
    return title;
}
- (void)setTitle:(NSString *)newTitle {
    if (title != newTitle) {
        [title release];
        title = [newTitle copy];
    }
}
// ....
```

```
- (NSString *)title {
    return title;
}
- (void)setTitle:(NSString *)newTitle
{
    if (title != newTitle) {
        [title release];
        title = [newTitle copy];
    }
}
// ....
```

```
@synthesize title;
```

```
@synthesize title = _title;
```

```
@synthesize title = _title;
```

Autosynthesis

[This slide intentionally left blank]

Private State

```
@interface Thing
{
   BOOL privateTest;
}

// ...
```

```
@interface Thing

@property BOOL privateTest;


// ...
```

```
@implementation Thing
{
   BOOL _privateTest;
}
// ...
```

```
@interface Thing ()

@property BOOL privateTest;

@end
// ...
```

```
@interface Thing (PrivateMethods)
- (void)doSomethingPrivate;
- (void)doSomethingElsePrivate;
@end
```

```
@interface Thing ()
- (void)doSomethingPrivate;
- (void)doSomethingElsePrivate;
@end
```

[This slide intentionally left blank]

Outlets

```
@interface MyViewController : MySuperclass {

    IBOutlet ElementClass *uiElement;
}

@end
```

```
@interface MyViewController : MySuperclass

@property (weak) IBOutlet ElementClass *uiElement;

@end
```

```
@interface MyViewController ()

@property (weak) IBOutlet ElementClass *uiElement;

@end
```

Initializer Methods and dealloc

```
- (id)init
{
    if (self = [super init]) {
        [self setTitle:@"default"];
    }
    return self;
}
```

```
- (id)init
{
    if ((self = [super init])) {
        [self setTitle:@"default"];
    }
    return self;
}
```

```objc
- (id)init
{
    self = [super init];
    if (self) {
        _title = @"default";
    }
    return self;
}
```

```
- (void)dealloc
{
    [self setTitle:nil];
    [super dealloc];
}
```

```
- (void)dealloc
{
    [_title release];
    [super dealloc];
}
```

[This slide intentionally left blank]

Protocols

Formalize communication channels

```
@ButlerProtocol

- (void)makeTea;
- (void)serveSandwiches;
- (void)mowTheLawn;

@end
```

```
@ButlerProtocol

- (void)makeTea;
- (void)serveSandwiches;
- (void)mowTheLawn;

@end




- (id <ButlerProtocol>)butler;
```

```
@interface NSObject (ButlerProtocol)

- (void)makeTea;
- (void)serveSandwiches;
- (void)mowTheLawn;

@end




- (id)butler;
```

```
@protocol ButlerProtocol <NSObject>

- (void)makeTea;
- (void)serveSandwiches;
@optional
- (void)mowTheLawn;

@end



@property id <ButlerProtocol> butler;
```

Collections and Literals

```
NSNumber *aNumber = [NSNumber numberWithFloat:2.3];

NSNumber *anotherNumber = [NSNumber numberWithFloat:x];

NSArray *anArray = [NSArray arrayWithObjects:aThing, @"A String",
                    [NSNumber numberWithFloat:3.14], nil];

NSDictionary *aDictionary = [NSDictionary dictionaryWithObjectsAndKeys:
                                value, @"Key",
                                [NSNumber numberWithBOOL:YES], @"OtherKey",
                                nil];
```

```
NSNumber *aNumber = @2.3f;

NSNumber *anotherNumber = @(x);

NSArray *anArray = @[ aThing, @"A String", @3.14 ];


NSDictionary *aDictionary = @{ @"Key" : value, @"OtherKey" : @YES };
```

```
NSDictionary *distanceDict = [NSDictionary dictionaryWithObjectsAndKeys:
            [NSNumber numberWithDouble:  0.0], kCIAttributeMin,
            [NSNumber numberWithDouble:  1.0], kCIAttributeMax,
            [NSNumber numberWithDouble:  0.0], kCIAttributeSliderMin,
            [NSNumber numberWithDouble:  0.7], kCIAttributeSliderMax,
            [NSNumber numberWithDouble:  0.2], kCIAttributeDefault,
            [NSNumber numberWithDouble:  0.0], kCIAttributeIdentity,
            kCIAttributeTypeScalar,            kCIAttributeType,
            nil];

NSDictionary *slopeDict = [NSDictionary dictionaryWithObjectsAndKeys:
            [NSNumber numberWithDouble: -0.01], kCIAttributeSliderMin,
            [NSNumber numberWithDouble:  0.01], kCIAttributeSliderMax,
            [NSNumber numberWithDouble:  0.00], kCIAttributeDefault,
            [NSNumber numberWithDouble:  0.00], kCIAttributeIdentity,
            kCIAttributeTypeScalar,            kCIAttributeType,
            nil];
```

```
NSDictionary *distanceDict = @{
        kCIAttributeMin       : @0.0,
        kCIAttributeMax       : @1.0,
        kCIAttributeSliderMin : @0.0,
        kCIAttributeSliderMax : @0.7,
        kCIAttributeDefault   : @0.2,
        kCIAttributeIdentity  : @0.0,
        kCIAttributeType      : kCIAttributeTypeScalar };


NSDictionary *slopeDict = @{
        kCIAttributeSliderMin : @-0.01,
        kCIAttributeSliderMax : @0.01,
        kCIAttributeDefault   : @0.00,
        kCIAttributeIdentity  : @0.00,
        kCIAttributeType      : kCIAttributeTypeScalar };
```

```
id firstElement = [anArray objectAtIndex:0];

[anArray replaceObjectAtIndex:0 withObject:newValue];
```

```
id firstElement = anArray[0];

anArray[0] = newValue;
```

```
id value = [aDictionary objectForKey:@"key"];

[aDictionary setObject:newValue forKey:@"key"];
```

```
id value = aDictionary[@"key"];

aDictionary[@"key"] = newValue;
```

```
NSArray *array = …;
int i;

for (i = 0; i < [array count]; i++) {

    id element = [array objectAtIndex:i];
    // …

}
```

```
NSArray *array = …;
int i;

for (i = 0; i < [array count]; i++) {

    id element = [array objectAtIndex:i];
    // …

}
```

```
NSArray *array = …;
int i;

for (i = 0; i < [array count]; i++) {

    id element = [array objectAtIndex:i];
    // …

}
```

```
NSArray *array = …;

for (id element in array) {

     // …

}
```

Blocks

```
NSArray *array = ...;
NSArray *sortedArray;
sortedArray = [array sortedArrayUsingFunction:MySort context:NULL];
```

```
NSInteger MySort(id num1, id num2, void *context)
{
    NSComparisonResult result;
    // Do comparison
    return result;
}
```

```
NSArray *array = ...;
BOOL reverse = ...;
NSArray *sortedArray;
sortedArray = [array sortedArrayUsingComparator:^(id num1, id num2) {
    NSComparisonResult result;
    // Do comparison
    return result;
}];
```

```
NSArray *array = …;

for (id element in array) {

     // …

}
```

```
NSArray *array = …;

[array enumerateObjectsUsingBlock:
      ^(id obj, NSUInteger idx, BOOL *stop) {
       // ...
       NSLog(@"Processing %@ at index %d”, obj, idx);
       // ...
}];
```

```objc
NSDictionary *dictionary = …;

for (NSString *key in dictionary) {

    id object = [dictionary objectForKey:key];

    // Do things with key and object.

}
```

```objc
NSDictionary *dictionary = …;

[dictionary enumerateKeysAndObjectsUsingBlock:^(id key, id object, BOOL *stop) {

    // Do things with key and object.

}];
```

```
- (void)registerForNotifications
{
    NSNotificationCenter *center = ...

    [center addObserver:self
        selector:@selector(windowBecameKey:)
        name:NSWindowDidBecomeKeyNotification
        object:self.window];
}

// Different context
// No queue information
- (void)windowBecameKey:(NSNotification *)notification
{
    // Get contextual information.
}
```

```
- (void)registerForNotifications
{
    NSNotificationCenter *center = ...

    [center addObserverForName:NSWindowDidBecomeKeyNotification
        object:self.window
        queue:[NSOperationQueue mainQueue]
        usingBlock:^(NSNotification *){
            // ...
            // Contextual information is already here
            // ...
    }];
}
```

```
- (void)registerForNotifications
{
    NSNotificationCenter *center = ...

    [center addObserverForName:NSWindowDidBecomeKeyNotification
        object:self.window
        queue:[NSOperationQueue mainQueue]
        usingBlock:^(NSNotification *) {
            // ...
            // Contextual information is already here
            // ...
    }];
}
```

```
- (void)registerForNotifications
{
    NSNotificationCenter *center = ...

    MyClass *__weak weakSelf = self;
    [center addObserverForName:NSWindowDidBecomeKeyNotification
        object:self.window
        queue:[NSOperationQueue mainQueue]
        usingBlock:^(NSNotification *) {
            // ...
            [weakSelf doSomething];
            // ...
    }];
}
```

Wrap Up

More Information

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

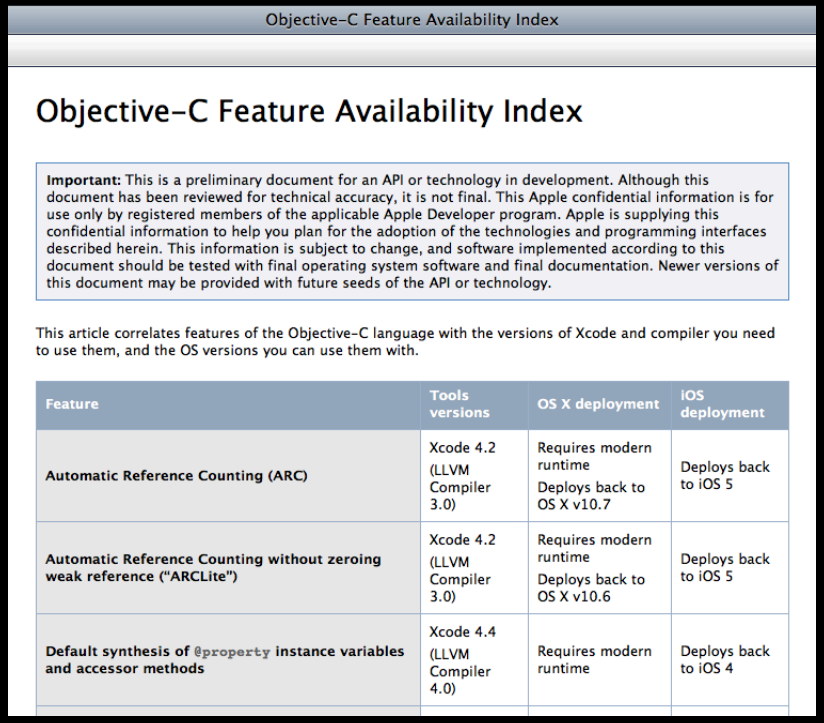
Apple Developer Forums
http://devforums.apple.com

Related Sessions

Modern Objective-C	Presidio Wednesday 10:15AM
Adopting Automatic Reference Counting	Marina Wednesday 11:30AM
Adopting Automatic Reference Counting (Repeat)	Nob Hill Friday 11:30AM

Documentation

- New document describes what features became available with which tools



Objective-C Feature Availability Index

Objective-C Feature Availability Index

Important: This is a preliminary document for an API or technology in development. Although this document has been reviewed for technical accuracy, it is not final. This Apple confidential information is for use only by registered members of the applicable Apple Developer program. Apple is supplying this confidential information to help you plan for the adoption of the technologies and programming interfaces described herein. This information is subject to change, and software implemented according to this document should be tested with final operating system software and final documentation. Newer versions of this document may be provided with future seeds of the API or technology.

This article correlates features of the Objective-C language with the versions of Xcode and compiler you need to use them, and the OS versions you can use them with.

Feature	Tools versions	OS X deployment	iOS deployment
Automatic Reference Counting (ARC)	Xcode 4.2 (LLVM Compiler 3.0)	Requires modern runtime Deploys back to OS X v10.7	Deploys back to iOS 5
Automatic Reference Counting without zeroing weak reference ("ARCLite")	Xcode 4.2 (LLVM Compiler 3.0)	Requires modern runtime Deploys back to OS X v10.6	Deploys back to iOS 5
Default synthesis of `@property` instance variables and accessor methods	Xcode 4.4 (LLVM Compiler 4.0)	Requires modern runtime	Deploys back to iOS 4

Summary

- New features make code more compact, expressive, correct, efficient
- Adopt them


WWDC2012