Modern Objective-C

Dave Koziol
Arbormoon Software, Inc.
@DaveKoziol
www.arbormoon.com

About Me

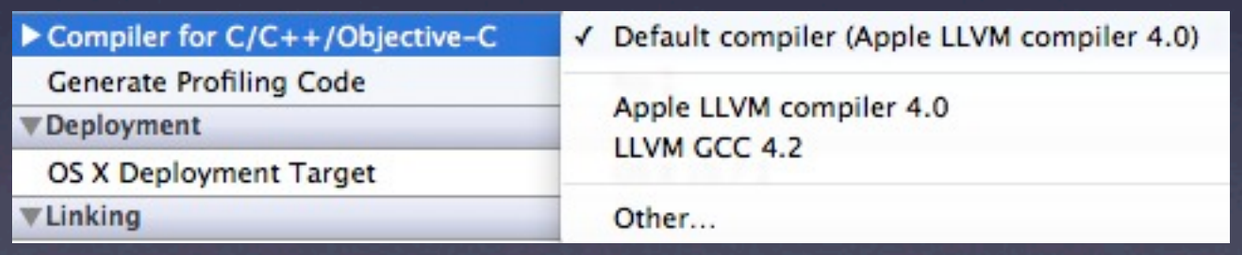
- Long time Apple Developer (21 WWDCs)
- Organizer Ann Arbor, MI CocoaHeads
- President & iOS Developer at Arbormoon Software Inc.
- Wunder Radio, KNBR, XanEdu, many other smaller apps.

Evolution of Objective-C

- Object-Oriented C
- Retain and Release
- Properties
- Blocks
- ARC

Requirements

- LLVM 4.0
- GCC is dead, get over it, and move on!
- Works on previous OSes.



Objective-C Popularity

- TIOBE Programming Community Index
- 2007 - 45th
- 2011 - 6th
- 2012 - 4th

Method Ordering

```
   @interface SongPlayer : NSObject
- (void)playSong:(Song *)song;
@end

@implementation SongPlayer
- (void)playSong:(Song *)song {
  NSError *error;
  [self startAudio:&error];
...
}

- (void)startAudio:(NSError **)error { ... }
@end
```

Enum with Fixed Underlying Type

```
typedef enum {
 NSNumberFormatterNoStyle,
 NSNumberFormatterDecimalStyle,
 NSNumberFormatterCurrencyStyle,
    NSNumberFormatterPercentStyle,
    NSNumberFormatterScientificStyle,
    NSNumberFormatterSpellOutStyle
} NSNumberFormatterStyle;
// typedef int NSNumberFormatterStyle;
```

- 32-bit and 64-bit portability issues

Enum with Fixed Underlying Type

```
enum {
 NSNumberFormatterNoStyle,
 NSNumberFormatterDecimalStyle,
 NSNumberFormatterCurrencyStyle,
 NSNumberFormatterPercentStyle,
 NSNumberFormatterScientificStyle,
 NSNumberFormatterSpellOutStyle
};
typedef NSUInteger NSNumberFormatterStyle;
```

- Pro: 32-bit and 64-bit portability
- Con: no formal relationship between type and enum constants

Enum with Fixed Underlying Type

```
typedef enum NSNumberFormatterStyle : NSUInteger {
 NSNumberFormatterNoStyle,
 NSNumberFormatterDecimalStyle,
 NSNumberFormatterCurrencyStyle,
 NSNumberFormatterPercentStyle,
 NSNumberFormatterScientificStyle,
 NSNumberFormatterSpellOutStyle
} NSNumberFormatterStyle;
```

- Better code completion
- Strong type checking
- 32 & 64 bit portable

Enum with Fixed Underlying Type

- NS_ENUM Macro

```
typedef NS_ENUM(NSUInteger, NSNumberFormatterStyle) {
    NSNumberFormatterNoStyle,
    NSNumberFormatterDecimalStyle,
    NSNumberFormatterCurrencyStyle,
    NSNumberFormatterPercentStyle,
    NSNumberFormatterScientificStyle,
    NSNumberFormatterSpellOutStyle
};
```

- Used by Foundation in Mac OS X 10.8

Enum with Fixed Underlying Type

- Stronger type checking (-Wconversion)

```
NSNumberFormatterStyle style = NSNumberFormatterRoundUp;
⚠ Implicit conversion from enumeration type 'enum NSNumberFormatterRoundingMode' to different enumeration type 'NSNumberFormatterStyle' (aka 'enum NSNumbe...
```

- Handling all enum values (-Wswitch)

```
switch (style)
{
⚠ 4 enumeration values not handled in switch: 'NSNumberFormatterDecimalStyle', 'NSNumberFormatterCurrencyStyle', 'NSNumberFormatterPercentStyle'
```

Property Synthesis

- Current Method

```
@interface Person : NSObject
@property(strong) NSString *name;
@end
```

```
@implementation Person
@synthesize name = _name;
@end
```

Property Synthesis

- New Hotness Method

```
@interface Person : NSObject
@property(strong) NSString *name;
@end
```

```
@implementation Person

@end
```

- Instance Variables are prefixed with “_”
- But backward compatible

Core Data NSManaged Object

- Opts out of synthesis by default
- Continue to use @property to declare typed accessors
- Continue to use @dynamic to inhibit warnings

New Literals

- NSNumber literals @42
- Boxed expression literals @(2+40)
-

Boxed String Expressions

- Boxed String Expressions
  @(getenv("PATH"))
- String Expression must be NULL Terminated
- String Expression must be UTF8
- Must not be NULL, or exception will be thrown.

Arrays

- Inconsistent Behavior

```
// if you write:
id a = nil, b = @"hello", c = @42;
NSArray *array = [NSArray arrayWithObjects:a, b, c, nil];

// You will get an empty array

// if you write:
id objects[] = { nil, @"hello", @42 };
NSUInteger count = sizeof(objects)/ sizeof(id);
array = [NSArray arrayWithObjects:objects count:count];

// An exception will be thrown
```

Array Literals

- NSArray @[], @[a, b]

```
// when you write this:
array = @[ a, b, c ];

// compiler generates:
id objects[] = { a, b, c };
NSUInteger count = sizeof(objects)/ sizeof(id);
array = [NSArray arrayWithObjects:objects count:count];

// An exception can be thrown
```

Dictionaries

- Potential for confusion/errors

```
NSDictionary *dict;

dict = [NSDictionary dictionaryWithObject:o1 forKey:k1];
```

```
dict = [NSDictionary dictionaryWithObjectsAndKeys:
                        o1, k1, o2, k2, o3, k3, nil];

id objects[] = { o1, o2, o3 };
id keys[] = { k1, k2, k3 };
NSUInteger count = sizeof(objects) / sizeof(id);
dict = [NSDictionary dictionaryWithObjects:objects
                                   forKeys:keys
                                     count:count];
```

Dictionary Literals

- NSDictionary @{}, @{k1:o1}, @{k1:o1, k2:o2}

```
// when you write this:
dict = @{ k1 : o1, k2 : o2, k3 : o3 };

// compiler generates:
id objects[] = { o1, o2, o3 };
id keys[] = { k1, k2, k3 };
NSUInteger count = sizeof(objects) / sizeof(id);
dict = [NSDictionary dictionaryWithObjects:objects
                                   forKeys:keys
                                     count:count];
```

Container Literals Restrictions

- All containers are immutable, use -mutablecopy
- Can’t set constants, recommend

```
static NSArray *thePlanets;
+ (void)initialize {
  if (self == [MyClass class]) {
    thePlanets = @[
      @"Mercury", @"Venus", @"Earth",
      @"Mars", @"Jupiter", @"Saturn",
      @"Uranus", @"Neptune"
    ];
  }
}
```

Container Subscripting

- NSArray songs[index]
- NSDictionary by index dictionary[@2]
- NSDictionary by object dictionary[@”Key”]

How Subscripting Works

```
    object[idx]
[object objectAtIndexedSubscript: idx];                 // Get
[object setObject: newObj atIndexedSubscript: idx];     // Set
```

```
object[key]
[object objectForKeyedSubscript:key];                   // Get
[object setObject: newObj forKeyedSubscript: key];      // Set
```

Your classes can be subscriptable

```
class MyList : NSObject
- (MyItem *)objectAtIndexedSubscript:(NSUInteger)idx;
- (void)setObject:(MyItem *)song atIndexedSubscript:(NSUInteger)idx;
@end

@implementation MyList {
   NSMutableArray *myItems;
}

- (MyItem *)objectAtIndexedSubscript:(NSUInteger)idx {
   return (MyItem *)myItems[idx];
}

- (void)setObject:(MyItem *)item atIndexedSubscript:(NSUInteger)idx {
   myItems[idx] = item;
}
```

ARC & CoreFoundation

- Implicit bridging via header files

  CF_IMPLICIT_BRIDGING_ENABLED

- Explicit ownership transform via cfbridgerelease

```
// Old Way
self.title = (__bridge NSString *)CFDictionaryGetValue(dict, @"title");

// New Way
self.title = (NSString *)CFDictionaryGetValue(dict, @"title");
```

C structs and ARC

- May not contain strong/weak object points
- Structs my be uninitialized
- Can be copied with memcpy
- Nothing explicit when they go out of scope.

C++ with ARC

- C++ structs can contain objects
- Blocks covert to C++ llambdas

Misc

- Mac OS X Garbage Collection is deprecated in Mountain Lion, use ARC Migrator.

Xcode 4.4

- Edit, Refactor, Convert to Object-C ARC
- Edit, Refactor, Convert to Modern Objective-C Syntax

Q&A