Modern Objective-C

Session 405

Patrick C. Beard
Objective-C Engineer

Objective-C is Popular

TIOBE Programming Community Index

2 0 0 7

45

RANK

Objective-C

Source: www.tiobe.com

2011

6

RANK

Objective-C

Source: www.tiobe.com

2012

4

RANK

Objective-C

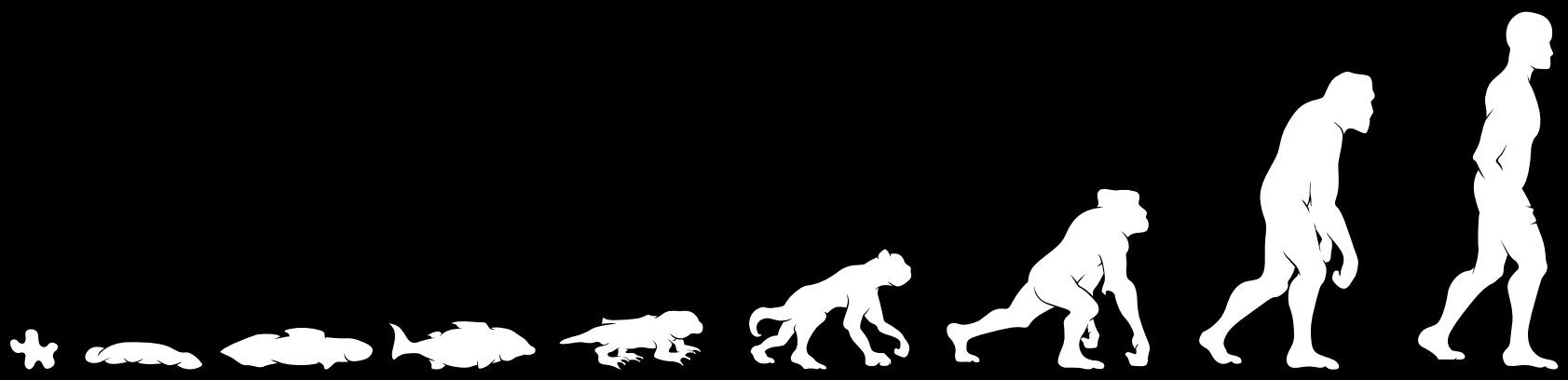
Source: www.tiobe.com

Evolution of Objective-C

Simpler, safer through automation

Evolution of Objective-C

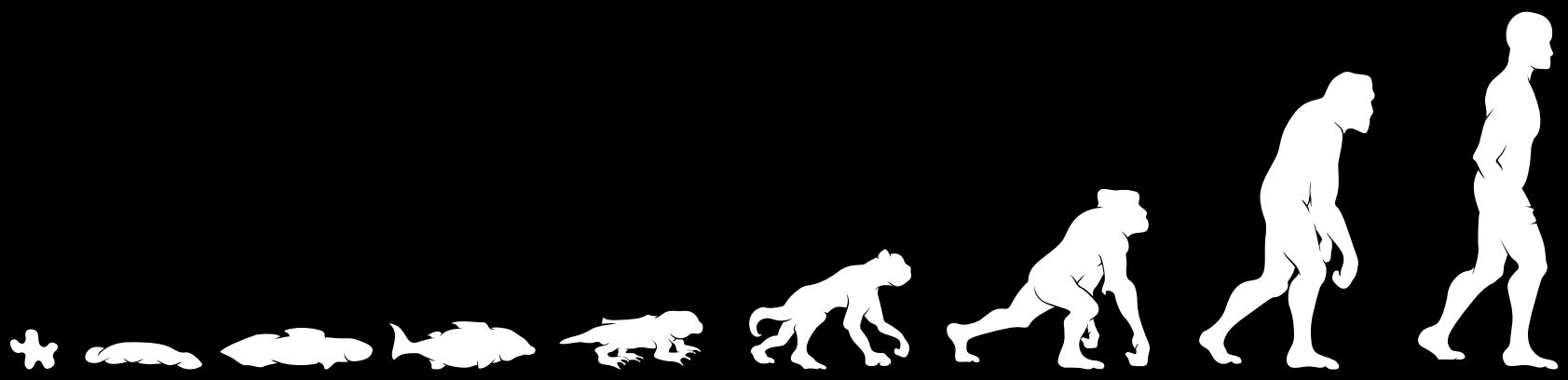
Simpler, safer through automation



Evolution of Objective-C

Simpler, safer through automation

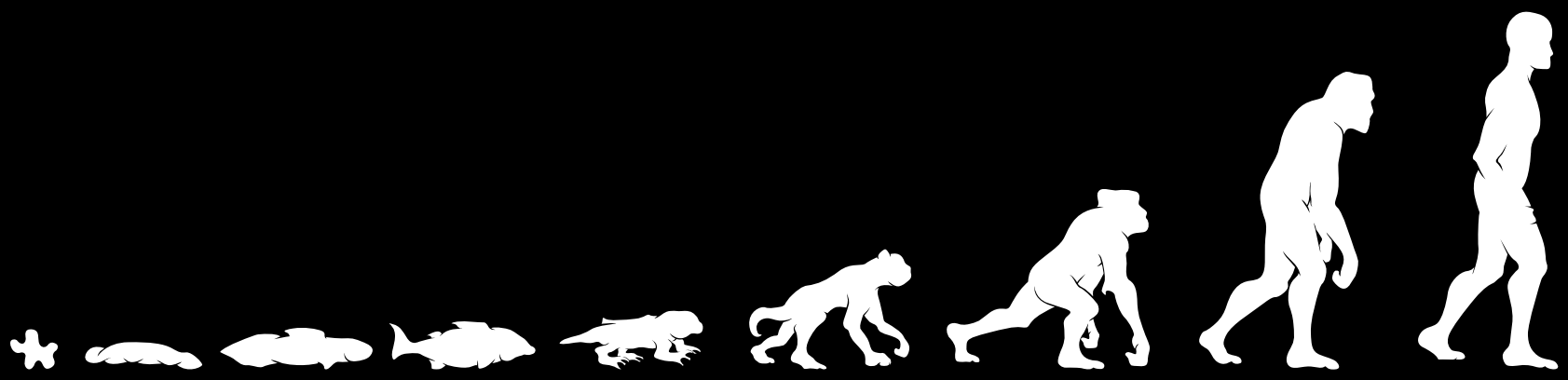
- Object-Oriented C



Evolution of Objective-C

Simpler, safer through automation

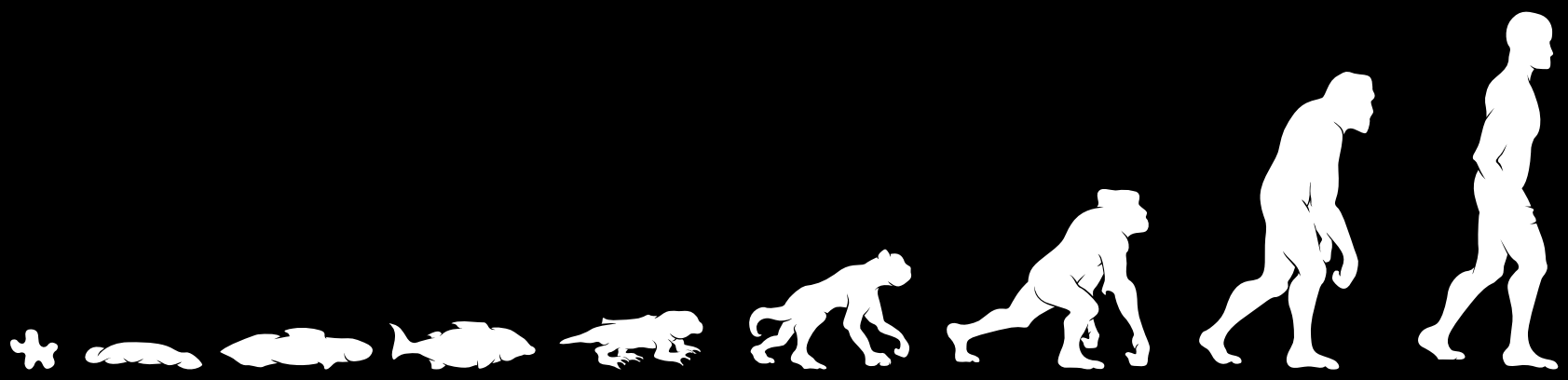
- Object-Oriented C
- Retain and Release



Evolution of Objective-C

Simpler, safer through automation

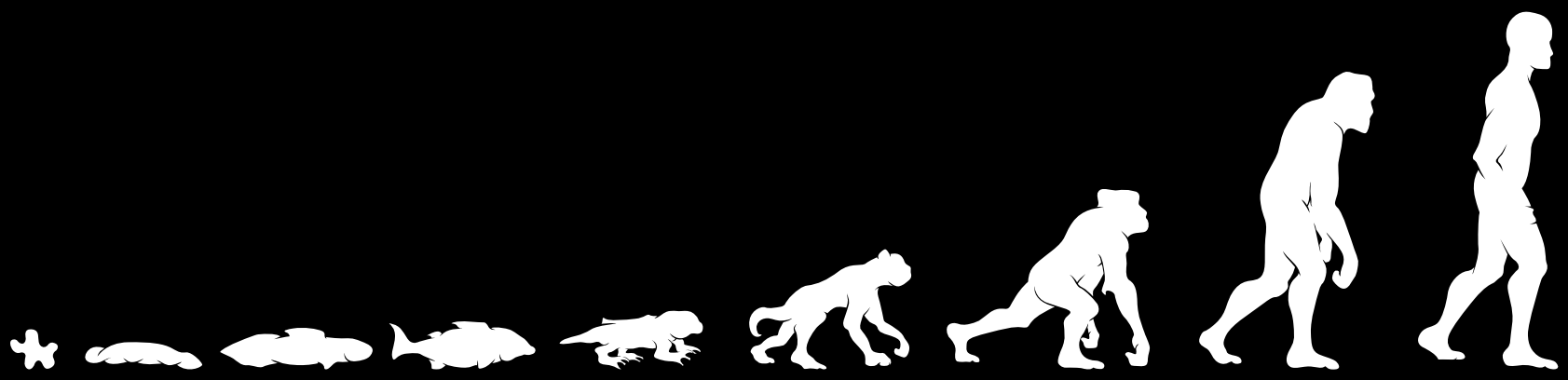
- Object-Oriented C
- Retain and Release
- Properties



Evolution of Objective-C

Simpler, safer through automation

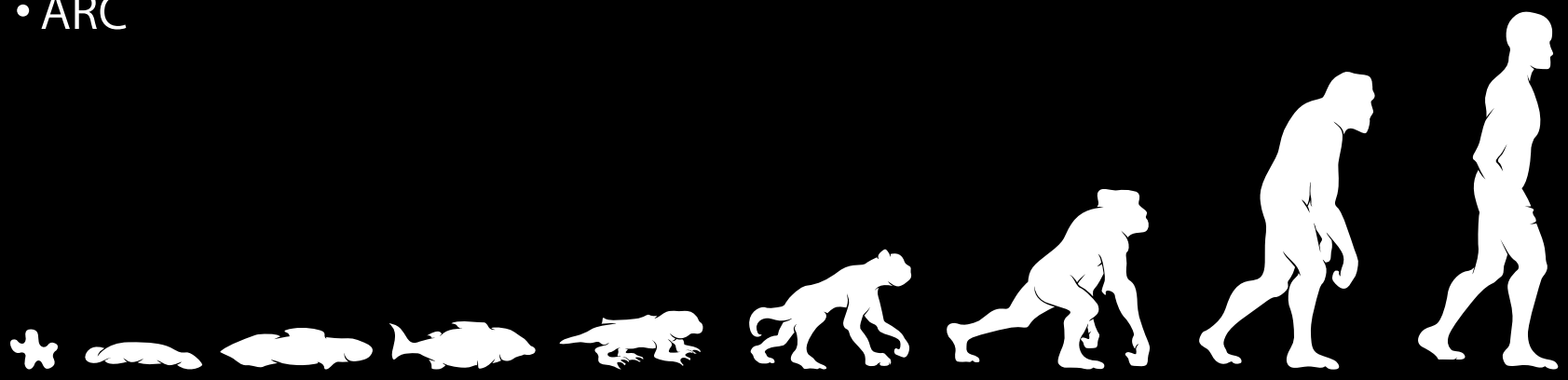
- Object-Oriented C
- Retain and Release
- Properties
- Blocks



Evolution of Objective-C

Simpler, safer through automation

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



What You Will Learn

Modern Objective-C

- New Objective-C language features
- Simplifying Objective-C code
- Avoiding common pitfalls

Method Ordering

Public and Private 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
```

Public and Private Method Ordering

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

@end

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

```
warning: instance method '-startAudio:' not found (return type defaults
to 'id')
    [self startAudio:&error];
    ^~~~~~~~~~~~~~~~~~~~~~~~
```

Wrong Workaround

In public interface

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

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

Okay Workaround

Class extension

```
@interface SongPlayer ()

- (void)startAudio:(NSError **)error;
@end

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

Okay Workaround

Class extension

```
@interface SongPlayer ()
@property(weak) IBOutlet UIButton *playButton;
- (void)startAudio:(NSError **)error;
@end

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

Okay Workaround Two

Reorder methods

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

@end

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

Best Solution

Just works…


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


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

Parsing the @implementation

Declarations then bodies


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

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

Parsing the @implementation

Declarations then bodies


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

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

Parsing the @implementation

Declarations then bodies


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

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

Parsing the @implementation

Declarations then bodies


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

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

Parsing the @implementation

Declarations then bodies


```
@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 Improvements

Enum with Indeterminate Type

Before OS X v10.5

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

Enum with Explicit Type

OS X v10.5 and iOS

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

Enum with Explicit Type

OS X v10.5 and iOS

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

- Pro: 32-bit and 64-bit portability

Enum with Explicit Type

OS X v10.5 and iOS

```
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

Xcode 4.4


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

- Better code completion
- Stronger type checking

Enum with Fixed Underlying Type

NS_ENUM macro


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

- Foundation declares like this

Enum with Explicit Type

Less helpful code completion

```
typedef NSUInteger NSAnimationCurve;
```

Requests the animation to run in the main thread in a custom run-loop mode that blocks user input. More...

```
NSNumb
_forma
K enum <anonymous> NSAnimationBlocking
V       NSString * NSAnimationDelayBinding
K enum <anonymous> NSAnimationEaseIn
K enum <anonymous> NSAnimationEaseInOut
K enum <anonymous> NSAnimationEaseOut
K enum <anonymous> NSAnimationEffectDisappearingItemDefault
K enum <anonymous> NSAnimationEffectPoof
K enum <anonymous> NSAnimationLinear
NSAnimationCurve curve = NSAnimationBlocking
```

Enum with Fixed Underlying Type

Better code completion

```
typedef enum NSNumberFormatterStyle : NSUInteger NSNumberFormatterStyle;
```

```
NSNumberFormatterStyle style = NSNumberFormatterSpellOutStyle
enum NSNumberFormatterRoundingMode NSNumberFormatterRoundFloor
enum NSNumberFormatterRoundingMode NSNumberFormatterRoundHalfD…
enum NSNumberFormatterRoundingMode NSNumberFormatterRoundHalfE…
enum NSNumberFormatterRoundingMode NSNumberFormatterRoundHalfUp
enum NSNumberFormatterRoundingMode NSNumberFormatterRoundUp
enum NSNumberFormatterStyle NSNumberFormatterScientific…
enum NSNumberFormatterStyle NSNumberFormatterSpellOutSt…
NSInteger NSNumberOfColorComponents(N…
Specifies a spell-out format; for example, “23” becomes “twenty-three”. More…
```

Enum with Fixed Underlying Type

Stronger type checking (-Wconversion)

```
NSNumberFormatterStyle style = NSNumberFormatterRoundUp;
```

Enum with Fixed Underlying Type

Stronger type checking (-Wconversion)

```
NSNumberFormatterStyle style = NSNumberFormatterRoundUp;
```

Enum with Fixed Underlying Type

Stronger type checking (-Wconversion)

```
NSNumberFormatterStyle style = NSNumberFormatterRoundUp;
```

Enum with Fixed Underlying Type

Stronger type checking (-Wconversion)

```
NSNumberFormatterStyle style = NSNumberFormatterRoundUp;
```

```
Implicit conversion from enumeration type
‘enum NSNumberFormatterRoundingMode’ to
different enumeration type
NSNumberFormatterPadPosition' to different enumeration type
'NSNumberFormatterStyle' (aka 'enum NSNumberFormatterStyle')
```

Enum with Fixed Underlying Type

Handling all enum values (-Wswitch)

```
void printStyle(NSNumberFormatterStyle style) {
  switch (style) {
  case NSNumberFormatterNoStyle:
    break;
  case NSNumberFormatterSpellOutStyle:
    break;
  }
}
```

Enum with Fixed Underlying Type

Handling all enum values (-Wswitch)

```
void printStyle(NSNumberFormatterStyle style) {
  switch (style) {
  case NSNumberFormatterNoStyle:
    break;
  case NSNumberFormatterSpellOutStyle:
    break;
  }
}
```

```
warning: 4 enumeration values not handled in switch:
'NSNumberFormatterDecimalStyle', 'NSNumberFormatterCurrencyStyle'
'NSNumberFormatterPercentStyle'...
```

Property Synthesis

Properties Simplify Classes

`@implementation` instance variables

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

@implementation Person {
    NSString *_name;
}



@synthesize name = _name;
@end
```

Properties Simplify Classes

`@implementation` instance variables

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

@implementation Person {
    NSString *_name;
}



@synthesize name = _name;
@end
```

Properties Simplify Classes

Synthesized instance variables

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

@implementation Person




@synthesize name = _name;
@end
```

Properties Simplify Classes

Synthesized instance variables

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

@implementation Person




@synthesize name = _name;
@end
```

Synthesize By Default

With Xcode 4.4 and later


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

@implementation Person




@end
```

Synthesize By Default

One line is all you need

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

@implementation Person
@end
```

Instance Variable Synthesis

What Is the Instance Variable Name?


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

@implementation Person


@end
```

What is the Instance Variable Name?

Instance variables now prefixed with "_"

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

@implementation Person
/* as if you'd written */
@synthesize name = _name;
@end
```

What is the Instance Variable Name?

Instance variables now prefixed with "_"

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

@implementation Person

- (NSString *)description {
  return _name;
}

@end
```

Synthesize by Default Rules

Readwrite properties

```
@property(readwrite) name;
```

-name	-setName:	ivar name
		_name
	√	_name
√		_name
√	√	

Synthesize by Default Rules

Readonly properties

```
@property(readonly) name;
```

-name	ivar name
	_name
√	

What About Backward Compatibility?

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

@implementation Person
/* what's the instance variable name? */
@synthesize name;
@end
```

`@synthesize` Is Backward Compatible

Be careful, when in doubt be fully explicit

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

@implementation Person
/* @synthesize name equivalent to */
@synthesize name = name;
@end
```

`@synthesize` Is Backward Compatible

Be careful, when in doubt be fully explicit

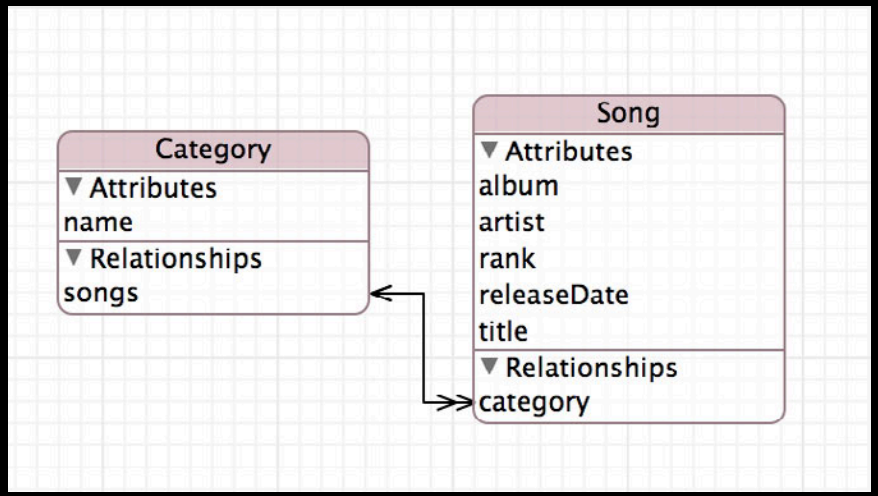
```
@interface Person : NSObject
@property(strong) NSString *name;
@end

@implementation Person

@synthesize name;

- (NSString *)description {
  return name;
}

@end
```

Core Data and Synthesize by Default

Model Attributes

Use @property in @interface

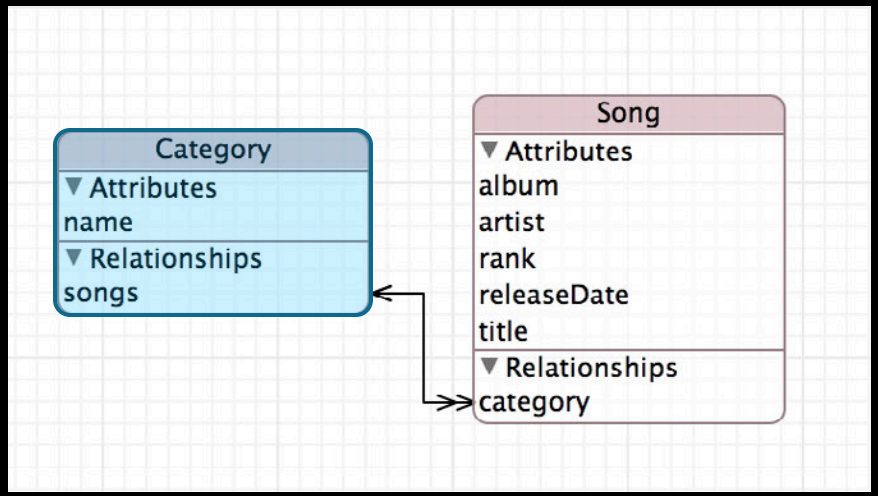
```
@interface Category : NSManagedObject
@property(strong) NSString *name;
@property(strong) NSSet *songs;
@end

@interface Song : NSManagedObject
@property(strong) NSString *album;
...
@property(strong) NSString *title;
@property(strong) Category *category;
@end
```



Model Attributes

Use @property in @interface

```
@interface Category : NSManagedObject
@property(strong) NSString *name;
@property(strong) NSSet *songs;
@end

@interface Song : NSManagedObject
@property(strong) NSString *album;
...
@property(strong) NSString *title;
@property(strong) Category *category;
@end
```

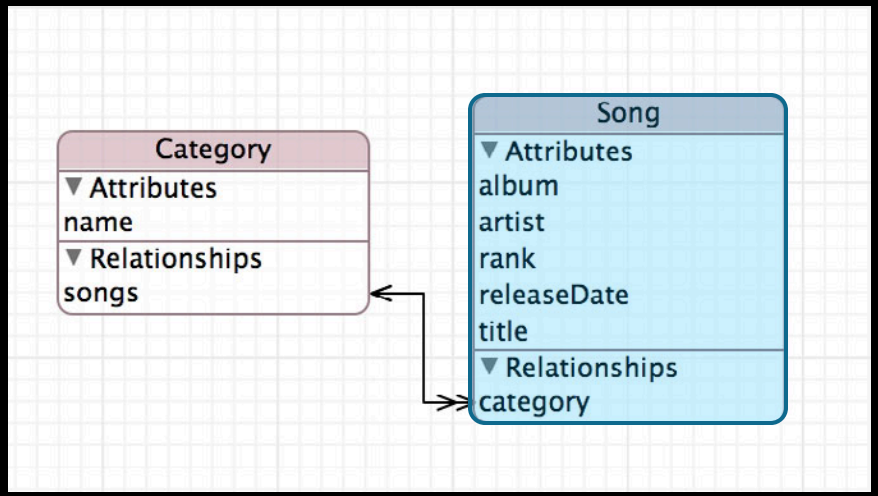


Model Attributes

Use @property in @interface

```
@interface Category : NSManagedObject
@property(strong) NSString *name;
@property(strong) NSSet *songs;
@end
```

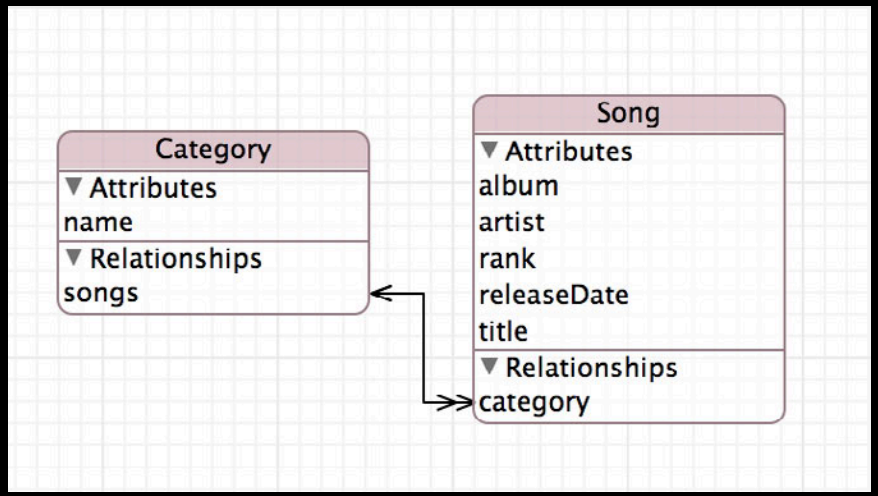
```
@interface Song : NSManagedObject
@property(strong) NSString *album;
...
@property(strong) NSString *title;
@property(strong) Category *category;
@end
```



Model Attributes

Use @dynamic in @implementation

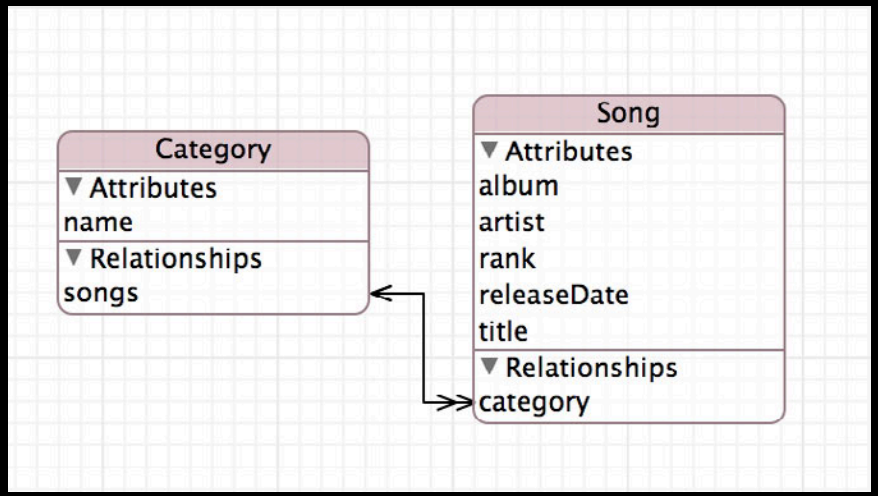
```
@implementation Category
@dynamic name;
@dynamic songs;
@end

@implementation Song
@dynamic album;
...
@dynamic title;
@dynamic category;
@end
```



Model Attributes

Use @dynamic in @implementation

```
@implementation Category
@dynamic name;
@dynamic songs;
@end

@implementation Song
@dynamic album;
...
@dynamic title;
@dynamic category;
@end
```

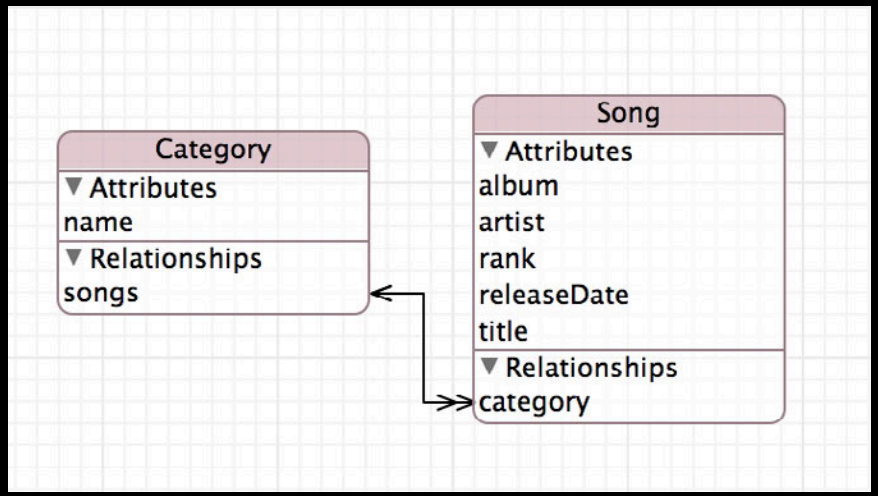


Model Attributes

Use @dynamic in @implementation

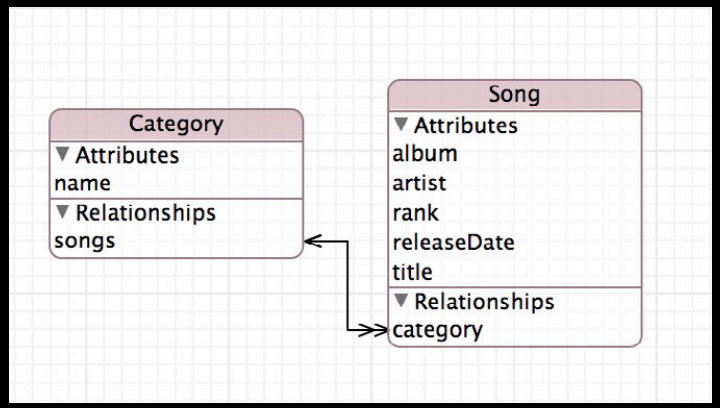
```
@implementation Category
@dynamic name;
@dynamic songs;
@end
```

```
@implementation Song
@dynamic album;
...
@dynamic title;
@dynamic category;
@end
```



Core Data NSManagedObject

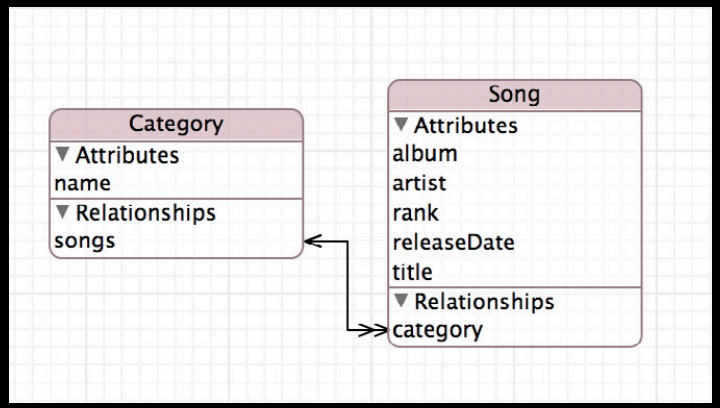
Opts out of synthesize by default



Core Data NSManagedObject

Opts out of synthesize by default

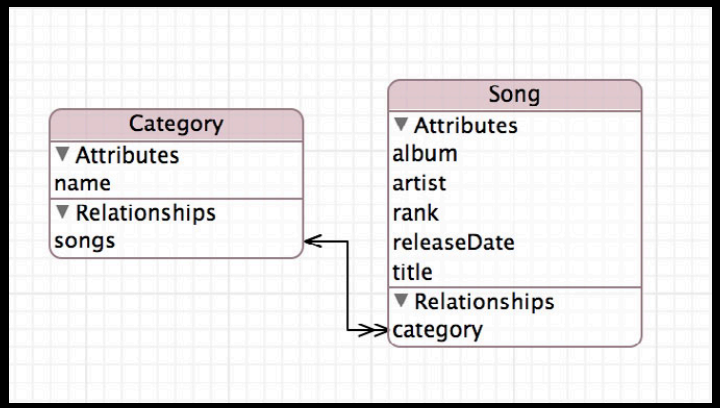
- NSManagedObject synthesizes properties



Core Data NSManagedObject

Opts out of synthesize by default

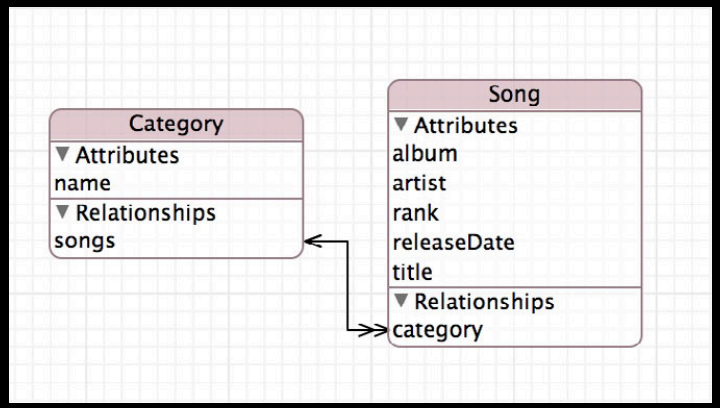
- NSManagedObject synthesizes properties
- Continue to use `@property` to declare typed accessors



Core Data NSManagedObject

Opts out of synthesize by default

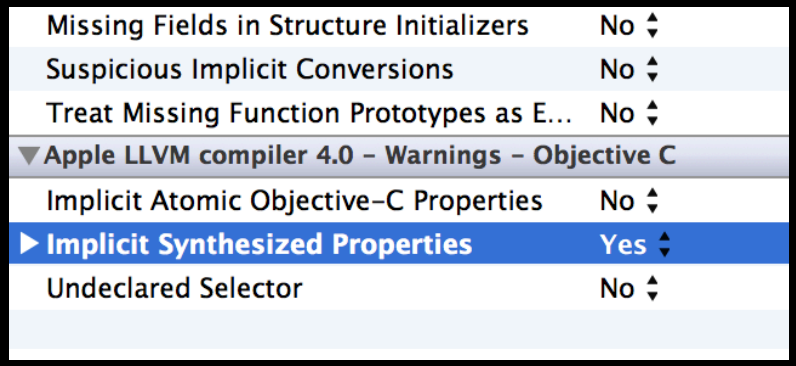
- NSManagedObject synthesizes properties
- Continue to use `@property` to declare typed accessors
- Continue to use `@dynamic` to inhibit warnings



Transitioning to Synthesize by Default

- If you use custom instance variable naming convention, enable this warning:

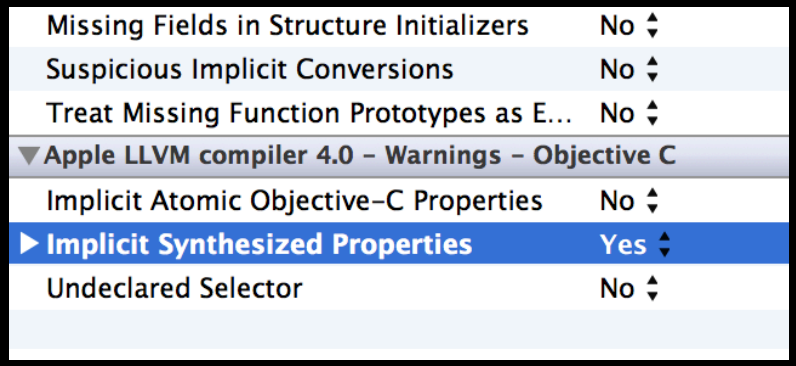
  `-Wobjc-missing-property-synthesis`



Transitioning to Synthesize by Default

- If you use custom instance variable naming convention, enable this warning:

```
-Wobjc-missing-property-synthesis
```



Synthesize by Default

Summary

- Each accessor method is synthesized unless explicitly provided
- Instance variable is generated unless all accessors provided
- `@synthesize` will always generate instance variable if not provided
- `@dynamic` inhibits all synthesis
- Core Data NSManagedObject subclasses opt out

Object Literals

NSNumber Literals

NSNumber Creation

```
NSNumber *value;

value = [NSNumber numberWithChar:'X'];

value = [NSNumber numberWithInt:12345];

value = [NSNumber numberWithUnsignedLong:12345ul];

value = [NSNumber numberWithLongLong:12345ll];

value = [NSNumber numberWithFloat:123.45f];

value = [NSNumber numberWithDouble:123.45];

value = [NSNumber numberWithBool:YES];
```

NSNumber Literals


```
NSNumber *value;

value = @'X';

value = [NSNumber numberWithInt:12345];

value = [NSNumber numberWithUnsignedLong:12345ul];

value = [NSNumber numberWithLongLong:12345ll];

value = [NSNumber numberWithFloat:123.45f];

value = [NSNumber numberWithDouble:123.45];

value = [NSNumber numberWithBool:YES];
```

NSNumber Literals


```
NSNumber *value;

value = @'X';

value = @12345;

value = [NSNumber numberWithUnsignedLong:12345ul];

value = [NSNumber numberWithLongLong:12345ll];

value = [NSNumber numberWithFloat:123.45f];

value = [NSNumber numberWithDouble:123.45];

value = [NSNumber numberWithBool:YES];
```

NSNumber Literals


```
NSNumber *value;

value = @'X';

value = @12345;

value = @12345ul;

value = [NSNumber numberWithLongLong:12345ll];

value = [NSNumber numberWithFloat:123.45f];

value = [NSNumber numberWithDouble:123.45];

value = [NSNumber numberWithBool:YES];
```

NSNumber Literals


```
NSNumber *value;

value = @'X';

value = @12345;

value = @12345ul;

value = @12345ll;

value = [NSNumber numberWithFloat:123.45f];

value = [NSNumber numberWithDouble:123.45];

value = [NSNumber numberWithBool:YES];
```

NSNumber Literals


```
NSNumber *value;

value = @'X';

value = @12345;

value = @12345ul;

value = @12345ll;

value = @123.45f;

value = [NSNumber numberWithDouble:123.45];

value = [NSNumber numberWithBool:YES];
```

NSNumber Literals


```
NSNumber *value;

value = @'X';

value = @12345;

value = @12345ul;

value = @12345ll;

value = @123.45f;

value = @123.45;

value = [NSNumber numberWithBool:YES];
```

NSNumber Literals


```
NSNumber *value;

value = @'X';

value = @12345;

value = @12345ul;

value = @12345ll;

value = @123.45f;

value = @123.45;

value = @YES;
```

Boxed Expression Literals

Expression Literals

```
NSNumber *piOverSixteen = [NSNumber numberWithDouble: ( M_PI / 16 )];

NSNumber *hexDigit = [NSNumber numberWithChar: "012345679ABCDEF"[i % 16]);

NSNumber *usesScreenFonts = [NSNumber numberWithBool:
                              [NSLayoutManager usesScreenFonts]];

NSNumber *writingDirection = [NSNumber numberWithInt:
                                NSWritingDirectionLeftToRight];

NSString *path = [NSString stringWithUTF8String: getenv("PATH")];
```

Expression Literals

```
NSNumber *piOverSixteen = @( M_PI / 16 );

NSNumber *hexDigit = [NSNumber numberWithChar: "012345679ABCDEF"[i % 16]];

NSNumber *usesScreenFonts = [NSNumber numberWithBool:
                              [NSLayoutManager usesScreenFonts]];

NSNumber *writingDirection = [NSNumber numberWithInt:
                                NSWritingDirectionLeftToRight];

NSString *path = [NSString stringWithUTF8String: getenv("PATH")];
```

Expression Literals

```
NSNumber *piOverSixteen = @( M_PI / 16 );

NSNumber *hexDigit = @( "012345679ABCDEF"[i % 16] );

NSNumber *usesScreenFonts = [NSNumber numberWithBool:
                             [NSLayoutManager usesScreenFonts]];

NSNumber *writingDirection = [NSNumber numberWithInt:
                               NSWritingDirectionLeftToRight];

NSString *path = [NSString stringWithUTF8String: getenv("PATH")];
```

Expression Literals

```
NSNumber *piOverSixteen = @( M_PI / 16 );

NSNumber *hexDigit = @( "012345679ABCDEF"[i % 16] );

NSNumber *usesScreenFonts = @( [NSLayoutManager usesScreenFonts] );


NSNumber *writingDirection = [NSNumber numberWithInt:
                                NSWritingDirectionLeftToRight];

NSString *path = [NSString stringWithUTF8String: getenv("PATH")];
```

Expression Literals

```
NSNumber *piOverSixteen = @( M_PI / 16 );

NSNumber *hexDigit = @( "012345679ABCDEF"[i % 16] );

NSNumber *usesScreenFonts = @( [NSLayoutManager usesScreenFonts] );

NSNumber *writingDirection = @( NSWritingDirectionLeftToRight );

NSString *path = [NSString stringWithUTF8String: getenv("PATH")];
```

Expression Literals

```
NSNumber *piOverSixteen = @( M_PI / 16 );

NSNumber *hexDigit = @( "012345679ABCDEF"[i % 16] );

NSNumber *usesScreenFonts = @( [NSLayoutManager usesScreenFonts] );


NSNumber *writingDirection = @( NSWritingDirectionLeftToRight );


NSString *path = @( getenv("PATH") );
```

Boxed String Expressions

```
NSString *path = [NSString stringWithUTF8String: getenv("PATH")];
for (NSString *dir in [path componentsSeparatedByString: @":"]) {
  // search for a file in dir...
}
```

Boxed String Expressions

```
NSString *path = @( getenv("PATH") );
for (NSString *dir in [path componentsSeparatedByString: @":"]) {
  // search for a file in dir...
}
```

Boxed String Expressions

```
NSString *path = @( getenv("PATH") );
for (NSString *dir in [path componentsSeparatedByString: @":"]) {
  // search for a file in dir...
}
```

- String expression must be '\0'-terminated, UTF8
- Must not be NULL, otherwise exception will be thrown

Container Literals

Container Literals

- Array Literals
- Dictionary Literals
- Scope

Array Literals

Array Creation

So many choices

```
NSArray *array;

array = [NSArray array];

array = [NSArray arrayWithObject:a];

array = [NSArray arrayWithObjects:a, b, c, nil];

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

Array Creation

So many choices

```
NSArray *array;

array = [NSArray array];

array = [NSArray arrayWithObject:a];

array = [NSArray arrayWithObjects:a, b, c, nil];

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

Array Creation

So many choices

```
NSArray *array;

array = [NSArray array];

array = [NSArray arrayWithObject:a];

array = [NSArray arrayWithObjects:a, b, c, nil];

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

Array Creation

So many choices

```
NSArray *array;

array = [NSArray array];

array = [NSArray arrayWithObject:a];

array = [NSArray arrayWithObjects:a, b, c, nil];

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

Array Creation

So many choices

```
NSArray *array;

array = [NSArray array];

array = [NSArray arrayWithObject:a];

array = [NSArray arrayWithObjects:a, b, c, nil];

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

Array Creation

So many choices

```
NSArray *array;

array = [NSArray array];

array = [NSArray arrayWithObject:a];

array = [NSArray arrayWithObjects:a, b, c, nil];

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

Nil Termination

Inconsistent behavior

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

Nil Termination

Inconsistent behavior

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

Array will be empty

Nil Termination

Inconsistent behavior

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

Array will be empty

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

Nil Termination

Inconsistent behavior

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

Array will be empty

```
// 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 Creation

Is there a better way?

```
NSArray *array;

array = [NSArray array];

array = [NSArray arrayWithObject:a];

array = [NSArray arrayWithObjects:a, b, c, nil];

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

Array Literals

Empty array

```
NSArray *array;

array = @[];

array = [NSArray arrayWithObject:a];

array = [NSArray arrayWithObjects:a, b, c, nil];

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

Array Literals

One element

```
NSArray *array;

array = @[];

array = @[ a ];

array = [NSArray arrayWithObjects:a, b, c, nil];

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

Array Literals

Multiple elements

```
NSArray *array;

array = @[];

array = @[ a ];

array = @[ a, b, c ];

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

Array Literals

Multiple elements

```
NSArray *array;

array = @[];

array = @[ a ];

array = @[ a, b, c ];

array = @[ a, b, c ];
```

How Array Literals Work

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

How Array Literals Work

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

How Array Literals Work

```
// 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];
```

Dictionary Literals

Dictionary Creation

More choices, and more chances for errors

```
NSDictionary *dict;

dict = [NSDictionary dictionary];

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 Creation

More choices, and more chances for errors

```
NSDictionary *dict;

dict = [NSDictionary dictionary];

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 Creation

More choices, and more chances for errors

```
NSDictionary *dict;

dict = [NSDictionary dictionary];

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 Creation

More choices, and more chances for errors

```
NSDictionary *dict;

dict = [NSDictionary dictionary];

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 Creation

More choices, and more chances for errors

```
NSDictionary *dict;

dict = [NSDictionary dictionary];

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 Creation

More choices, and more chances for errors

```
NSDictionary *dict;

dict = [NSDictionary dictionary];

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

Empty dictionary

```
NSDictionary *dict;

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

One element

```
NSDictionary *dict;

dict = @{};

dict = @{ k1 : o1 };

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

One element

```
NSDictionary *dict;

dict = @{};

dict = @{ k1 : o1 };

dict = @{ k1 : o1, k2 : o2, k3 : o3 };


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

Multiple elements

```
NSDictionary *dict;

dict = @{};

dict = @{ k1 : o1 };

dict = @{ k1 : o1, k2 : o2, k3 : o3 };
```

```
dict = @{ k1 : o1, k2 : o2, k3 : o3 };
```

How Dictionary Literals Work

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

How Dictionary Literals Work

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

How Dictionary Literals Work

```
// 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 Literal Restrictions

All Containers Immutable

- Simply use `-mutableCopy`

```
NSMutableArray *mutablePlanets = [@[
  @"Mercury", @"Venus", @"Earth",
  @"Mars", @"Jupiter", @"Saturn",
  @"Uranus", @"Neptune"
] mutableCopy];
```

What About Constant Containers

```
static NSArray *
thePlanets = @[
  @"Mercury", @"Venus", @"Earth",
  @"Mars", @"Jupiter", @"Saturn",
  @"Uranus", @"Neptune"
];
```

What About Constant Containers

```
static NSArray *
thePlanets = @[
  @"Mercury", @"Venus", @"Earth",
  @"Mars", @"Jupiter", @"Saturn",
  @"Uranus", @"Neptune"
];
```

What About Constant Containers

```
static NSArray *
thePlanets = @[
  @"Mercury", @"Venus", @"Earth",
  @"Mars", @"Jupiter", @"Saturn",
  @"Uranus", @"Neptune"
];
```

error: array literals not constant

Workaround

Use `+initialize` method

```
@implementation MyClass

static NSArray *thePlanets;

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

Workaround

Use `+initialize` method

```
@implementation MyClass

static NSArray *thePlanets;

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

No Constant Dictionaries

- Compile time hashing/sorting
- Framework dictionary keys not constant

No Constant Dictionaries

- Compile time hashing/sorting
- Framework dictionary keys not constant

```
NSString * const key = NSFilePosixPermissions;
```

No Constant Dictionaries

- Compile time hashing/sorting
- Framework dictionary keys not constant

```
NSString * const key = NSFilePosixPermissions;
```

```
static NSDictionary *encryptedFileAttributes = @{
  NSFileProtectionKey : NSFileProtectionComplete,
  NSFilePosixPermissions : @0700
};
```

No Constant Dictionaries

- Compile time hashing/sorting
- Framework dictionary keys not constant

```
NSString * const key = NSFilePosixPermissions;
```

```
static NSDictionary *encryptedFileAttributes = @{
  NSFileProtectionKey : NSFileProtectionComplete,
  NSFilePosixPermissions : @0700
};
```

```
error: initializer is not a compile-time constant
```

Object Subscripting

Array Subscripting

```
@implementation SongList {
  NSMutableArray *_songs;
}

- (Song *)replaceSong:(Song *)newSong atIndex:(NSUInteger)idx {
    Song *oldSong = [_songs objectAtIndex:idx];
    [_songs replaceObjectAtIndex:idx withObject:newSong];
    return oldSong;
}
```

Array Subscripting


```
@implementation SongList {
  NSMutableArray *_songs;
}

- (Song *)replaceSong:(Song *)newSong atIndex:(NSUInteger)idx {
    Song *oldSong = _songs[idx];
    _songs[idx] = newSong;
    return oldSong;
}
```

Dictionary Subscripting

```
@implementation Database {
  NSMutableDictionary *_storage;
}

- (id)replaceObject:(id)object forKey:(id <NSCopying>)key {
    id oldObject = [_storage objectForKey:key];
    [_storage setObject:object forKey:key];
    return oldObject;
}
```

Dictionary Subscripting


```
@implementation Database {
  NSMutableDictionary *_storage;
}

- (id)replaceObject:(id)newObject forKey:(id <NSCopying>)key {
    id oldObject = _storage[key];
    _storage[key] = newObject;
    return oldObject;
}
```

How Subscripting Works

How Subscripting Works

- Indexed subscripting

```
_songs[idx] = newSong;
```

How Subscripting Works

- Indexed subscripting

```
_songs[idx] = newSong;
```

- Keyed subscripting

```
_fileAttributes[NSFilePosixPermissions] = @0700;
```

Indexed Subscripting

```
- (Song *)replaceSong:(Song *)newSong atIndex:(NSUInteger)idx {
    Song *oldSong = _songs[idx];
    _songs[idx] = newSong;
    return oldSong;
}
```

Indexed Subscripting

```
- (Song *)replaceSong:(Song *)newSong atIndex:(NSUInteger)idx {
    Song *oldSong = [_songs objectAtIndexedSubscript:idx];
    _songs[idx] = newSong;
    return oldSong;
}
```

Indexed Subscripting

```
- (Song *)replaceSong:(Song *)newSong atIndex:(NSUInteger)idx {
    Song *oldSong = [_songs objectAtIndexedSubscript:idx];
    [_songs setObject:newSong atIndexedSubscript:idx];
    return oldSong;
}
```

Keyed Subscripting

```
- (id)replaceObject:(id)newObject forKey:(id <NSCopying>)key {
    id oldObject = _storage[key];
    _storage[key] = newObject;
    return oldObject;
}
```

Keyed Subscripting

```
- (id)replaceObject:(id)newObject forKey:(id <NSCopying>)key {
    id oldObject = [_storage objectForKeyedSubscript:key];
    _storage[key] = newObject;
    return oldObject;
}
```

Keyed Subscripting

```
- (id)replaceObject:(id)newObject forKey:(id <NSCopying>)key {
    id oldObject = [_storage objectForKeyedSubscript:key];
    [_storage setObject:newObject forKeyedSubscript:key];
    return oldObject;
}
```

Subscripting Methods

- Indexed subscripting methods

```
- (elementType)objectAtIndexedSubscript:(indexType)idx;
- (void)setObject:(elementType)object
        atIndexedSubscript:(indexType)idx;
```

- Keyed subscripting methods

```
- (elementType)objectForKeyedSubscript:(keyType)key;
- (void)setObject:(elementType)object
        forKeyedSubscript:(keyType)key;
```

- `indexType` must be integral
- `elementType` and `keyType` must be an object pointer

Your Classes Can Be Subscriptable

```
class MyGroovySongList : NSObject
- (Song *)objectAtIndexedSubscript:(NSUInteger)idx;
- (void)setObject:(Song *)song atIndexedSubscript:(NSUInteger)idx;
@end
@implementation MyGroovySongList {
  NSMutableArray *mySongs;
}
- (Song *)objectAtIndexedSubscript:(NSUInteger)idx {
  return (Song *)mySongs[idx];
}
- (void)setObject:(Song *)song atIndexedSubscript:(NSUInteger)idx {
  mySongs[idx] = song;
}
```

Your Classes Can Be Subscriptable

```
class MyGroovySongList : NSObject
- (Song *)objectAtIndexedSubscript:(NSUInteger)idx;
- (void)setObject:(Song *)song atIndexedSubscript:(NSUInteger)idx;
@end
@implementation MyGroovySongList {
  NSMutableArray *mySongs;
}
- (Song *)objectAtIndexedSubscript:(NSUInteger)idx {
  return (Song *)mySongs[idx];
}
- (void)setObject:(Song *)song atIndexedSubscript:(NSUInteger)idx {
  mySongs[idx] = song;
}
```

Your Classes Can Be Subscriptable

```
class MyGroovySongList : NSObject
- (Song *)objectAtIndexedSubscript:(NSUInteger)idx;
- (void)setObject:(Song *)song atIndexedSubscript:(NSUInteger)idx;
@end
@implementation MyGroovySongList {
  NSMutableArray *mySongs;
}
- (Song *)objectAtIndexedSubscript:(NSUInteger)idx {
  return (Song *)mySongs[idx];
}
- (void)setObject:(Song *)song atIndexedSubscript:(NSUInteger)idx {
  mySongs[idx] = song;
}
```

Your Classes Can Be Subscriptable

```
class MyGroovySongList : NSObject
- (Song *)objectAtIndexedSubscript:(NSUInteger)idx;
- (void)setObject:(Song *)song atIndexedSubscript:(NSUInteger)idx;
@end
@implementation MyGroovySongList {
  NSMutableArray *mySongs;
}
- (Song *)objectAtIndexedSubscript:(NSUInteger)idx {
  return (Song *)mySongs[idx];
}
- (void)setObject:(Song *)song atIndexedSubscript:(NSUInteger)idx {
  mySongs[idx] = song;
}
```

Availability

Availability

- Xcode 4.4 and later
  - `@synthesize` by default
  - Unordered method declarations
  - Enums with fixed underlying type
  - Subscripting and object literals, boxed expressions
- Results work on previous OS releases

Demo

Convert to modern syntax

Using Objective-C++ with ARC

C structs under ARC

May not contain strong/weak object pointers

C structs under ARC

May not contain strong/weak object pointers

```
struct Event {
  id             what;
  NSPoint        where;
  NSTimeInterval when;
};
```

C structs under ARC

May not contain strong/weak object pointers

```
struct Event {
  id              what; error: Objective-C objects not allowed in struct or union
  NSPoint         where;
  NSTimeInterval when;
};
```

C structs under ARC

May not contain strong/weak object pointers

```
struct Event {
  id              what; error: Objective-C objects not allowed in struct or union
  NSPoint         where;
  NSTimeInterval when;
};
```

- Why prohibit?

C structs under ARC

May not contain strong/weak object pointers

```
struct Event {
  id              what; error: Objective-C objects not allowed in struct or union
  NSPoint         where;
  NSTimeInterval when;
};
```

- Why prohibit?
  - structs may be uninitialized (contain garbage values)

C structs under ARC

May not contain strong/weak object pointers

```
struct Event {
  id              what; error: Objective-C objects not allowed in struct or union
  NSPoint         where;
  NSTimeInterval when;
};
```

- Why prohibit?
  - structs may be uninitialized (contain garbage values)
  - Can be copied with memcpy() skipping ownership transfer

C structs under ARC

May not contain strong/weak object pointers

```
struct Event {
  id              what; error: Objective-C objects not allowed in struct or union
  NSPoint         where;
  NSTimeInterval when;
};
```

- Why prohibit?
  - structs may be uninitialized (contain garbage values)
  - Can be copied with memcpy() skipping ownership transfer
  - Nothing happens when they go out of scope

Solution One

Convert to Objective-C class

```
@interface Event : NSObject {
@private
  id what;
  NSPoint where;
  NSTimeInterval when;
}
@end
```

Solution One

Convert to Objective-C class

```
@interface Event : NSObject

@property(strong) id what;
@property NSPoint where;
@property NSTimeInterval when;

@end
```

Solution Two

Objective-C++ structs may contain object pointers

```
struct Event {
  id             what;
  NSPoint        where;
  NSTimeInterval when;
};
```

Objective-C++ struct

With ARC managed instance variables

```
struct Event {
  id             what;
  NSPoint        where;
  NSTimeInterval when;
  Event() ... {}
  ~Event() { ... }
  Event(const Event &e) : ... {}
  Event& operator=(const Event &e) { ... }
  Event(Event &&old) ... { ... }
  Event& operator=(Event &&old) { ... }
};
```

Objective-C++ struct

With ARC managed instance variables

```
struct Event {
  id             what;
  NSPoint        where;
  NSTimeInterval when;
  Event() ... {}
  ~Event() { ... }
  Event(const Event &e) : ... {}
  Event& operator=(const Event &e) { ... }
  Event(Event &&old) ... { ... }
  Event& operator=(Event &&old) { ... }
};
```

Implicitly Generated
Compiler implicitly generates these

Default Constructor

Initializes object pointers to nil

```
struct Event {
  id             what;
  NSPoint        where;
  NSTimeInterval when;
  Event() : what(nil) {}
};
```

Destructor

Releases object pointers

```
struct Event {
  id             what;
  NSPoint        where;
  NSTimeInterval when;
  Event() : what(nil) {}
  ~Event() { objc_release(what); }
};
```

Copy Constructor and Assignment Operator

Retains object pointers

```
struct Event {
  id             what;
  NSPoint        where;
  NSTimeInterval when;
  Event() : what(nil) {}
  ~Event() { objc_release(what); }
  Event(const Event &e) : what(objc_retain(e.what), ... {}
  Event& operator=(const Event &e) {
    objc_release(what), what = objc_retain(e.what); ...
  }
};
```

C++11 Move Constructor and Operator

Steals object pointers

```
struct Event {
  id             what;
  NSPoint        where;
  NSTimeInterval when;
  Event() : what(nil) {}
  ~Event() { objc_release(what); }
  Event(const Event &e) : ... { ... }
  Event& operator=(const Event &e) { ... }
  Event(Event &&old) : what(old.what), ... { old.what = nil; }
  Event& operator=(Event &&old) {
    objc_release(what), what = old.what, that.what = nil; ...
  }
};
```

Objective-C++ struct

With ARC managed instance variables

```
struct Event {
  id             what;
  NSPoint        where;
  NSTimeInterval when;
  Event() ... {}
  ~Event() { ... }
  Event(const Event &e) : ... {}
  Event& operator=(const Event &e) { ... }
  Event(Event &&old) ... { ... }
  Event& operator=(Event &&old) { ... }
};
```

Objective-C++ struct

With ARC managed instance variables

```
struct Event {
  id             what;
  NSPoint        where;
  NSTimeInterval when;
  Event() ... {}
  ~Event() { ... }
  Event(const Event &e) : ... {}
  Event& operator=(const Event &e) { ... }
  Event(Event &&old) ... { ... }
  Event& operator=(Event &&old) { ... }
};
```

Implicitly Generated
Compiler implicitly generates these

Objective-C++ Vector

With ARC `__strong` pointers

Objective-C++ Vector

With ARC `__strong` pointers

```
@implementation MyWindow
```

Objective-C++ Vector

With ARC `__strong` pointers

```
@implementation MyWindow
typedef void (^view_visitor_t)(NSView *view);
```

Objective-C++ Vector

With ARC `__strong` pointers

```
@implementation MyWindow
typedef void (^view_visitor_t)(NSView *view);
```

Objective-C++ Vector

With ARC `__strong` pointers

```
@implementation MyWindow
typedef void (^view_visitor_t)(NSView *view);

- (void)visitAllViews:(view_visitor_t)visitor {
  std::vector<NSView *> views;
  views.push_back(self.contentView);
  while (views.size()) {
    NSView *view = views.back();
    views.pop_back();
    visitor(view);
    for (NSView *subview in view.subviews)
      views.push_back(subview);
  }
}
```

Strong Ownership
Inserting object pointer into container retains

Objective-C++ Vector

With ARC `__strong` pointers

```
@implementation MyWindow
typedef void (^view_visitor_t)(NSView *view);

- (void)visitAllViews:(view_visitor_t)visitor {
  std::vector<NSView *> views;
  views.push_back(self.contentView);
  while (views.size()) {
    NSView *view = views.back();
    views.pop_back();
    visitor(view);
    for (NSView *subview in view.subviews)
      views.push_back(subview);
  }
}
```

Strong Ownership
Removing object pointer from container releases

Compacting the Vector

Using a C++11 lambda

```
- (void)compact {
  _observers.erase(std::remove_if(_observers.begin(), _observers.end(),
                                  [](id o) { return o == nil; }),
                   known_widgets.end());
}
```

Compacting the Vector

Using a C++11 lambda

```
- (void)compact {
  _observers.erase(std::remove_if(_observers.begin(), _observers.end(),
                                  [](id o) { return o == nil; }),
                   known_widgets.end());
}
```

Compacting the Vector

Or use a block, compiler wraps it in a lambda

```
- (void)compact {
  _observers.erase(std::remove_if(_observers.begin(), _observers.end(),
                                  ^(id o) { return o == nil; }),
                   known_widgets.end());
}
```

ARC and Core Foundation

Explicit Ownership Transfer

Old way

- Needed for C functions with Copy or Create in the name, which are inferred to return +1 retained references

```
NSArray *people = (__bridge_transfer NSArray *)
                   ABAddressBookCopyPeopleWithName(addressBook,
                                                   CFSTR("Appleseed"));
```

Explicit Ownership Transfer

Recommended way

```
NSArray *people = CFBridgingRelease(
                    ABAddressBookCopyPeopleWithName(addressBook,
                                                    CFSTR("Appleseed")));
```

Explicit Ownership Transfer

Recommended way

```
NSArray *people = CFBridgingRelease(
                    ABAddressBookCopyPeopleWithName(addressBook,
                                                    CFSTR("Appleseed")));
```

Explicit Bridging

```
@property(copy) CFDictionaryRef properties;

- (CFDictionaryRef)properties {
  CFMutableDictionaryRef dict = CFDictionaryCreateMutable(...);
  CFDictionaryAddValue(dict, @"title", (__bridge CFStringRef)self.title);
  ...
  return dict;
}

- (void)setProperties:(CFDictionaryRef)properties {
  self.title = (__bridge NSString *)CFDictionaryGetValue(dict, @"title");
  ...
}
```

Implicit Bridging


```
CF_IMPLICIT_BRIDGING_ENABLED
...

CF_EXPORT
const void *CFDictionaryGetValue(CFDictionaryRef theDict, const void *key);

CF_EXPORT
void CFDictionaryAddValue(CFMutableDictionaryRef theDict, const void *key,
const void *value);

...
CF_IMPLICIT_BRIDGING_DISABLED
```

Using Implicit Bridging

No ownership transfer needed

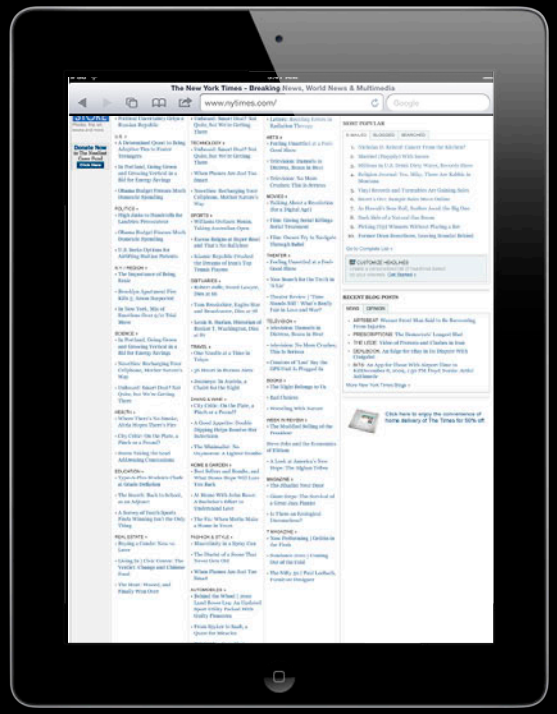

```
@property(readonly) CFDictionaryRef properties;

- (CFDictionaryRef)properties {
  CFMutableDictionaryRef dict = CFDictionaryCreateMutable(...);
  CFDictionaryAddValue(dict, @"title", (CFStringRef)self.title);
  ...
  return dict;
}

- (void)setProperties:(CFDictionaryRef)properties {
  self.title = (NSString *)CFDictionaryGetValue(dict, @"title");
  ...
}
```

Garbage Collection

- Superseded by ARC
- Deprecated in Mountain Lion
- ARC migrator can help

Garbage Collection

- Superseded by ARC
- Deprecated in Mountain Lion
- ARC migrator can help



Garbage Collection

- Superseded by ARC
- Deprecated in Mountain Lion
- ARC migrator can help

More Information

Michael Jurewitz
Developer Tools Evangelist
jury@apple.com

Documentation
LLVM Compiler Language Extensions
http://clang.llvm.org/docs/LanguageExtensions.html

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

Related Sessions

Adopting Automatic Reference Counting	Marina Wednesday 11:30AM
Migrating to Modern Objective-C	Nob Hill Thursday 3:15PM

Labs

Objective-C and Automatic Reference Counting Lab	Developer Tools Lab A Wednesday 2:00PM
Objective-C and Automatic Reference Counting Lab	Developer Tools Lab C Thursday 2:00PM

Summary

- `@synthesize` by default
- Forward declarations optional
- Fixed underlying type enums
- Literals and subscripting
  - Boxed expressions
- GC is deprecated

WWDC2012