What's New in Foundation for iOS 4

Chris Kane
Cocoa Frameworks

Lots of Changes

- Major update to Foundation
  - iPhone OS 2 and 3 ⇔ Mac OS X 10.5
  - iOS 4 ⇔ Mac OS X 10.6
- Covering highlights here
  - Blocks and new block-using APIs
  - Other changes

Blocks

What Are Blocks?

- A block is an object which represents a chunk of code
  - Similar to “closures” or “function objects”
- Blocks are objects
  - Respond to NSObject methods
- Available to C, C++, Objective C, and Objective C++ code

Blocks

Snippets of code used like a function pointer

Declaring a variable "myBlock"
The "^" declares this to be a block

This is a block literal

```
int (^myBlock)(int) = ^(int num) { return num * multiplier; };
```

myBlock is a block that returns an int

It takes a single argument, also an int

The argument is named num

This is the body of the block

Block Capture

Blocks can access and capture scope

What is this ???

```
int multiplier = 7;
int (^myBlock)(int) = ^(int num) { return num * multiplier; };
multiplier = 13;

printf("%d\n", myBlock(3));
```

Has no effect on the block

Calling myBlock like a function

```
⇒ 21
```

Blocks

Used as an argument to an Objective-C method

Method taking a block argument

Argument name

```
@interface NSArray
- (void)enumerateObjectsUsingBlock:
          (void (^)(id object, NSUInteger index, BOOL *stop))block;
@end
```

Argument type

Blocks

Used as an argument to an Objective-C method

```
@interface NSArray
- (void)enumerateObjectsUsingBlock:
          (void (^)(id object, NSUInteger index, BOOL *stop))block;
@end
```

Argument 1 (id)

Argument 2 (NSUInteger)

Block return type (none here)

Argument 3 (pointer to BOOL)

Using enumerateObjectsUsingBlock:

```
NSString *nameToSearchFor = …;
[myArray enumerateObjectsUsingBlock:
    ^(id object, NSUInteger index, BOOL *stop) {
        if ([[object name] isEqual:nameToSearchFor]) {
            NSLog(@“Found object at index %ld”, index);
        }
    }];
```

Writing to Local Variables

```
id found = nil;
NSString *nameToSearchFor = ...;
[myArray enumerateObjectsUsingBlock:
    ^(id object, NSUInteger index, BOOL *stop) {
        if ([[object name] isEqual:nameToSearchFor]) {
            found = object;    compile error
            *stop = YES;
        }
    }];
if (found != nil) ...
```

Writing to Local Variables

```
__block id found = nil;
NSString *nameToSearchFor = ...;
[myArray enumerateObjectsUsingBlock:
    ^(id object, NSUInteger index, BOOL *stop) {
        if ([[object name] isEqual:nameToSearchFor]) {
            found = object;
            *stop = YES;
        }
    }];
if (found != nil) ...
```

New Collection APIs Using Blocks

- Various methods on
  - NSArray
  - NSDictionary
  - NSSet
  - NSIndexSet
- Enumeration
  - Invoke a block for each object in a collection

Naming Patterns

```
@interface NSArray
- (void)enumerateObjectsUsingBlock:
        (void (^)(id object, NSUInteger index, BOOL *stop))block;
@end

@interface NSDictionary
- (void)enumerateKeysAndObjectsUsingBlock:
        (void (^)(id key, id object, BOOL *stop))block;
@end
```

Naming Patterns

```
@interface NSArray
- (void)enumerateObjectsWithOptions:(NSUInteger)options usingBlock:…

- (void)enumerateObjectsAtIndexes:(NSIndexSet *)indexes options:…
usingBlock:…
@end
```

Options

Enumerate a subset,
for ordered collections

Enumeration Options

```
enum {
    NSEnumerationReverse,
    NSEnumerationConcurrent
};
```

New Collection APIs Using Blocks

Searching

```
- (NSIndexSet *)indexesOfObjectsPassingTest:
                    (BOOL (^)(…))predicate;
```

NSArray's version returns an index set

Returns a boolean

- Enumeration options apply to these methods as well

Blocks

Used in a new typedef

```
typedef NSComparisonResult (^NSComparator)(id obj1, id obj2);
```

New Collection APIs Using Blocks

Sorting

```
@interface NSArray
- (NSArray *)sortedArrayUsingComparator:(NSComparator)comparator;
@end


NSArray *sortedArray = [myArray sortedArrayUsingComparator:
        ^(id str1, id str2) {
            return [str1 localizedStandardCompare:str2];
        }
    ];
```

Sorting Options

```
enum {
    NSSortConcurrent,
    NSSortStable
};
```

Binary Search

Binary search in a sorted array

```
- (NSUInteger)indexOfObject:(id)obj
                inSortedRange:(NSRange)r
                      options:(NSBinarySearchingOptions)opts
             usingComparator:(NSComparator)cmp;
```

NSString Enumerating

Strings can also be enumerated

```
- (void)enumerateSubstringsInRange:(NSRange)range
                          options:(NSStringEnumerationOptions)opts
                       usingBlock:(void (^)(NSString *substring,
                                            NSRange substringRange,
                                            NSRange enclosingRange,
                                            BOOL *stop))block;
```

- Can enumerate by lines, paragraphs, words, sentences

Regular Expressions

NSRegularExpression

Represents a regular expression

```
NSRegularExpression *regEx;
regEx = [NSRegularExpression regularExpressionWithPattern:@“..d”
                                                  options:0
                                                    error:&err];
```

NSRegularExpression

Finding matches

```
NSString *str = @"good food today";
NSRange r = NSMakeRange(0, [str length]);
NSArray *array = [regEx matchesInString:str options:0 range:r];

   ..d  ..d ..d
⇒ good food today
```

- Matches are represented by NSTextCheckingResult objects
  - Overall range and ranges of capture groups

NSRegularExpression

Enumerating matches

```
- (void)enumerateMatchesInString:(NSString *)string
                        options:(NSMatchingOptions)options
                          range:(NSRange)range
                     usingBlock:(void (^)(NSTextCheckingResult *result,
                                          NSMatchingFlags flags,
                                          BOOL *stop))block;
```

NSRegularExpression

Replacing

```
NSString *res = [regEx stringByReplacingMatchesInString:str
                                                options:0
                                                  range:r
                                           withTemplate:@"#"];
```

```
str = @“good food today”
⇒ @“g# f# #ay”
```

NSString and Regular Expressions

- New -rangeOfString:… option in iPhone OS 3.2 and 4
  - NSRegularExpressionSearch
  - Search string is treated as a regular expression
- New option also applies to find-and-replace methods as of iOS 4

Advanced Text Handling for iPhone OS	Nob Hill Tuesday 4:30PM

Other Changes

Delegate Protocols

Old way: category on NSObject

```
@interface NSObject (NSXMLParserDelegate)
- (void)parserDidStartDocument:(NSXMLParser *)parser;
- (void)parserDidEndDocument:(NSXMLParser *)parser;
…
@end
```

Delegate Protocols Now

New way: formal protocols

```
@protocol NSXMLParserDelegate <NSObject>
@optional
- (void)parserDidStartDocument:(NSXMLParser *)parser;
- (void)parserDidEndDocument:(NSXMLParser *)parser;
…
@end
```

- Delegate methods updated

```
- (id <NSXMLParserDelegate>)delegate;
- (void)setDelegate:(id <NSXMLParserDelegate>)delegate;
```

NSPropertyListSerialization

- New methods which produce NSErrors
- New stream-based methods

```
NSError *err;
id plist = [NSPropertyListSerialization
                propertyListWithStream:stream
                               options:NSPropertyListImmutable
                                format:NULL
                                 error:&err];
if (nil == plist) {
    // look at and deal with the NSError
}
```

NSData

Searching

```
- (NSRange)rangeOfData:(NSData *)dataToFind
              options:(NSDataSearchOptions)mask
                range:(NSRange)searchRange;

enum {
    NSDataSearchBackwards,
    NSDataSearchAnchored
};
```

NSString

New compare method

```
- (NSComparisonResult)localizedStandardCompare:(NSString *)string;
```

- Performs the "system standard" string comparison
  - Use for presenting sorted lists/tables in the UI
  - Exact behavior may change between OS versions

NSAttributedString

- A string, plus sets of attributes that apply to parts of the string
- Added in iPhone OS 3.2

Green text color attribute

This text is green and **bold**

Bold font attribute and green color

NSFileWrapper

- File wrappers represent a file-system node as an object
- Typically used to create and manage "wrapper" directories
  - A directory of files to be treated as a single entity
  - Useful to group core document file with other "assets", like images
- Offers fast multipart document saving

NSOperation

- New waiting method

```
- (void)waitUntilFinished;
```

- Have a block run after the operation finishes

```
- (void (^)(void))completionBlock;
- (void)setCompletionBlock:(void (^)(void))block;
```

NSBlockOperation

- New subclass of NSOperation

```
+ (id)blockOperationWithBlock:(void (^)(void))block
- (void)addExecutionBlock:(void (^)(void))block
```

- Multiple blocks are executed concurrently

NSOperationQueue

- Now implemented on top of GCD
- New methods

```
- (NSUInteger)operationCount
- (void)addOperations:(NSArray *)ops waitUntilFinished:(BOOL)wait;
```

- Adding a block to an operation queue

```
- (void)addOperationWithBlock:(void (^)(void))block;
```

- New special queues

```
+ (NSOperationQueue *)currentQueue
+ (NSOperationQueue *)mainQueue
```

NSNotificationCenter

New method to add a block as an observer

```
- (id)addObserverForName:(NSString *)name
                  object:(id)obj
                   queue:(NSOperationQueue *)queue
              usingBlock:(void (^)(NSNotification *))block;
```

NSNotificationCenter

New method to add a block as an observer

Observer

```
- (id)addObserverForName:(NSString *)name
                  object:(id)obj
                   queue:(NSOperationQueue *)queue
              usingBlock:(void (^)(NSNotification *))block;
```

- Return value is retained by the system
- Use -removeObserver: with return value to unregister later

NSNotificationCenter

New method to add a block as an observer

```
- (id)addObserverForName:(NSString *)name
                  object:(id)obj
                   queue:(NSOperationQueue *)queue
              usingBlock:(void (^)(NSNotification *))block;
```

Queue

- Block will be run on the queue
- Queue causes asynchronous handling
- Posting still waits for all observer handlers to finish

NSDateFormatter

Given a set of components, get a localized format string

```
NSString *format = [NSDateFormatter
                    dateFormatFromTemplate:@“Mdjm”
                                   options:0
                                    locale:[NSLocale currentLocale]];
```

⇒ “M/d h:mm a”

```
[dateFormatter setDateFormat:format];
NSString *string = [dateFormatter stringFromDate:[NSDate date]];
```

⇒ “6/8 11:02 AM”

NSDateFormatter

Relative date naming

- Formatting with “Yesterday”, “Today”, “Tomorrow”

```
- (BOOL)doesRelativeDateFormatting;
- (void)setDoesRelativeDateFormatting:(BOOL)b;
```

```
⇒ “Yesterday 11:04 AM”
⇒ “Today 11:04 AM”
⇒ “Tomorrow 11:04 AM”
```

NSCache

A new class for caching

- Keep around expensive objects
- Entries may be discarded automatically on memory pressure
- Dictionary-like API

```
- (id)objectForKey:(id)key;
- (void)setObject:(id)obj forKey:(id)key;
```

- NSCaches are thread-safe

Summary

Wrap-Up

- Blocks and new block-taking APIs
- Various other changes
- See documentation for more info

Related Sessions

What's New in Cocoa Touch (Repeat)	Marina Friday 11:30AM
Advanced Text Handling for iPhone OS	Nob Hill Tuesday 4:30PM
Working Effectively With Objective-C in iPhone OS	Pacific Heights Wednesday 9:00AM
Introducing Blocks and Grand Central Dispatch on iPhone	Russian Hill Wednesday 11:30AM
Future Proofing Your Application	Pacific Heights Thursday 2:00PM
API Design for Cocoa and Cocoa Touch	Marina Thursday 4:30PM

Labs

Cocoa Touch Lab	App Frameworks Lab D Tuesday 2:00PM
Cocoa Lab	App Frameworks Lab C Tuesday 3:15PM
Cocoa Lab	App Frameworks Lab D Thursday 2:00PM
Application Compatibility Lab	App Frameworks Lab B Thursday 4:30PM

WWDC10