iOS Application Development

Foundation Classes

Ashiq Uz Zoha (Ayon),
ashiq.ayon@gmail.com
Dhrubok Infotech Services Ltd.

Foundation Framework

- Value and collection classes
- User defaults
- Archiving
- Notifications
- Undo manager
- Tasks, timers, threads
- File system, pipes, I/O, bundles

NSObject

- Root class
- Implements many basics
  - Memory management
  - Introspection
  - Object equality

NSString

- General-purpose Unicode string support
  - Unicode is a coding system which represents all of the world's languages
- Consistently used throughout Cocoa Touch instead of "char *"
- Without doubt the most commonly used class
- Easy to support any language in the world with Cocoa

String Constants

- In C constant strings are

```
"simple"
```

- In ObjC, constant strings are

```
@"just as simple"
```

- Constant strings are NSString instances

```
NSString *aString = @"Hello World!";
```

Format Strings

- Similar to printf, but with %@ added for objects

```
NSString *aString = @"Johnny";
NSString *log = [NSString stringWithFormat: @"It's '%@'", aString];
```

log would be set to `It's 'Johnny'`

- Also used for logging

```
NSLog(@"I am a %@, I have %d items", [array className], [array count]);
```

would log something like:

```
I am a NSArray, I have 5 items
```

NSString

- Often ask an existing string for a new string with modifications

```
- (NSString *)stringByAppendingString:(NSString *)string;
- (NSString *)stringByAppendingFormat:(NSString *)string;
- (NSString *)stringByDeletingPathComponent;
```

- Example:

```
NSString *myString = @"Hello";
NSString *fullString;
fullString = [myString stringByAppendingString:@" world!"];
```

fullString would be set to `Hello world!`

NSString

- Common NSString methods

```
- (BOOL)isEqualToString:(NSString *)string;
- (BOOL)hasPrefix:(NSString *)string;
- (int)intValue;
- (double)doubleValue;
```

- Example:

```
NSString *myString = @"Hello";
NSString *otherString = @"449";
if ([myString hasPrefix:@"He"]) {
      // will make it here
}
if ([otherString intValue] > 500) {
      // won't make it here
}
```

NSMutableString

- NSMutableString subclasses NSString
- Allows a string to be modified
- Common NSMutableString methods

```
+ (id)string;
- (void)appendString:(NSString *)string;
- (void)appendFormat:(NSString *)format, ...;
```

```
NSMutableString *newString = [NSMutableString string];
[newString appendString:@"Hi"];
[newString appendFormat:@", my favorite number is: %d",
   [self favoriteNumber]];
```

Collections

- Array - ordered collection of objects
- Dictionary - collection of key-value pairs
- Set - unordered collection of unique objects
- Common enumeration mechanism
- Immutable and mutable versions
  - Immutable collections can be shared without side effect
  - Prevents unexpected changes
  - Mutable objects typically carry a performance overhead

NSArray

- Common NSArray methods

```
+ arrayWithObjects:(id)firstObj, ...;  // nil terminated!!!
- (unsigned)count;
- (id)objectAtIndex:(unsigned)index;
- (unsigned)indexOfObject:(id)object;
```

- NSNotFound returned for index if not found

```
NSArray *array = [NSArray arrayWithObjects:@"Red", @"Blue",
@"Green", nil];
if ([array indexOfObject:@"Purple"] == NSNotFound) {
      NSLog (@"No color purple");
}
```

- Be careful of the nil termination!!!

NSMutableArray

- NSMutableArray subclasses NSArray
- So, everything in NSArray
- Common NSMutableArray Methods

```
+ (NSMutableArray *)array;
- (void)addObject:(id)object;
- (void)removeObject:(id)object;
- (void)removeAllObjects;
- (void)insertObject:(id)object atIndex:(unsigned)index;

NSMutableArray *array = [NSMutableArray array];
[array addObject:@"Red"];
[array addObject:@"Green"];
[array addObject:@"Blue"];
[array removeObjectAtIndex:1];
```

NSDictionary

- Common NSDictionary methods

```
+ dictionaryWithObjectsAndKeys: (id)firstObject, ...;
- (unsigned)count;
- (id)objectForKey:(id)key;
```

- nil returned if no object found for given key

```
NSDictionary *colors = [NSDictionary
      dictionaryWithObjectsAndKeys:@"Red", @"Color 1",
      @"Green", @"Color 2", @"Blue", @"Color 3", nil];
NSString *firstColor = [colors objectForKey:@"Color 1"];


if ([colors objectForKey:@"Color 8"]) {
    // won't make it here
}
```

NSMutableDictionary

- NSMutableDictionary subclasses NSDictionary
- Common NSMutableDictionary methods

```
+ (NSMutableDictionary *)dictionary;
- (void)setObject:(id)object forKey:(id)key;
- (void)removeObjectForKey:(id)key;
- (void)removeAllObjects;
```

```
NSMutableDictionary *colors = [NSMutableDictionary dictionary];

[colors setObject:@"Orange" forKey:@"HighlightColor"];
```

NSSet

- Unordered collection of objects
- Common NSSet methods

```
+ setWithObjects:(id)firstObj, ...;    // nil terminated
- (unsigned)count;
- (BOOL)containsObject:(id)object;
```

NSMutableSet

- NSMutableSet subclasses NSSet
- Common NSMutableSet methods

```
+ (NSMutableSet *)set;
- (void)addObject:(id)object;
- (void)removeObject:(id)object;
- (void)removeAllObjects;
- (void)intersectSet:(NSSet *)otherSet;
- (void)minusSet:(NSSet *)otherSet;
```

Enumeration

- Consistent way of enumerating over objects in collections
- Use with NSArray, NSDictionary, NSSet, etc.

```
NSArray *array = ... ; // assume an array of People objects

// old school
Person *person;
int count = [array count];
for (i = 0; i < count; i++) {
    person = [array objectAtIndex:i];
    NSLog([person description]);
}

// new school
for (Person *person in array) {
   NSLog([person description]);
}
```

NSNumber

- In Objective-C, you typically use standard C number types
- NSNumber is used to wrap C number types as objects
- Subclass of NSValue
- No mutable equivalent!
- Common NSNumber methods

```
+ (NSNumber *)numberWithInt:(int)value;
+ (NSNumber *)numberWithDouble:(double)value;
- (int)intValue;
- (double)doubleValue;
```

Other Classes

- NSData / NSMutableData
  - Arbitrary sets of bytes
- NSDate / NSCalendarDate
  - Times and dates

Getting some objects

- Until we talk about memory management:
  - Use class factory methods
    - NSString's +stringWithFormat:
    - NSArray's +array
    - NSDictionary's +dictionary
  - Or any method that returns an object except alloc/init or copy.

More ObjC Info?

- http://developer.apple.com/documentation/Cocoa/Conceptual/ObjectiveC
- Concepts in Objective C are applicable to any other OOP language

Source :

- http://www.stanford.edu/class/cs193p/cgi-bin/drupal/node/21
- http://developer.apple.com

Thanks…