iOS App Development

1.3 Foundation Framework

Foundation Framework

- Base Layer of Objective-C Classes
- Primitive Object Classes
- Collection Classes
- A lot more ...

Foundation Data Types

- id: Pointer to an Object
- BOOL: YES or NO
- NSInteger/NSUInteger
- NSTimeInterval

Primitive Object Classes

- NSObject: Base super class
- NSString: Unicode String Literal
- NSNumber: int, BOOL, double, ...
- NSValue*: NSRange, CGPoint, ...
- NSData (binary data)
- NSDate

NSString

- Unicode String
- Immutable - Use NSMutableString instead
- Primitive methods:
  - - length
  - - characterAtIndex:
- Native Support for Objective-C String Literal:
  - NSString *string = @”String”;
- Character Encoding Support

Collection Classes

- NSArray: Ordered Collection of Objects
- NSSet/NSOrderedSet: Collection of Distinct Objects
- NSDictionary: Associations of Keys and Values

Mutability

- NSArray/NSDictionary are immutable collection of objects
- NSMutableArray/NSMutableDictionary are mutable
  - Collection can be modified
- NSMutableString, NSMutableSet, NSMutableURLRequest, ...

NSArray/ NSMutableArray

- NSArray *array = @[@"One", @(YES), @(3.14)];

Before Modern Objective-C

```
NSArray *array = [[NSArray alloc] initWithObjects: @"A", @"B", nil];

object = [array objectAtIndex:0];

...

NSMutableArray *marray = ...;

[marray replaceObjectAtIndex:1 withObject:@"Two"];
```

Array Enumeration

```
NSArray *myArray = @[ @"1.0", @"2.0", @"3.0"];
for (NSString *string in myArray) {
        double value = [string doubleValue];
        NSLog(@"%2.1f", value);
}

// or, Using ‘block’
[myArray enumerateObjectsUsingBlock:
  ^(id obj, NSUInteger idx, BOOL *stop) {
    double value = [(NSString *)obj doubleValue];
    NSLog(@"%2.1f", value);
  }];
```

Array Sorting

```
NSInteger intSort(id num1, id num2, void *context) {
    int v1 = [num1 intValue];
    int v2 = [num2 intValue];
    if (v1 < v2)
        return NSOrderedAscending;
    else if (v1 > v2)
        return NSOrderedDescending;
    else
        return NSOrderedSame;
}

- (void) sort {
    NSArray *array = @[ @(3), @(1), @(10)]; // array of NSNumber objects
    NSArray *sortedArray = [array sortedArrayUsingFunction:intSort
context:NULL];
}
```

NSDictionary/ NSMutableDictionary

- `NSDictionary *map = @{ @"name" : @"John", @"married" : @(NO)];`
- `name = map[@"name"];`
- `NSMutableDictionary *mutable = [map mutableCopy];`
- `mutable[@"married"] = @(YES);`

Set and Ordered Set

- NSSet/NSMutableSet
  - Unordered collection of distinct objects
- NSOrderedSet/NSMutableOrderedSet
  - Ordered collection of distinct objects
  - Faster to check “contains” than an array.

Property List

- Collection of collections: Property List의 조건
  - NSArray, NSDictionary, NSNumber, NSString, NSDate, NSData로만 구성된 object graph
- NSArray나 NSDictionary가 Property List이면 관련 API를 통해서 File에 쓰거나 읽는것이 수월하다
  - 이를 위해서는 NSArray의 내용이나 NSDictionary의 key와 value가 모두 Property List 조건을 만족해야 한다.
- `[plist writeToFile:@”filename” atomically:YES]; // plist is NSArray or NSDictionary`

NSUserDefaults

- NSUserDefaults class

Sample methods:

```
- (void)setDouble:(double)double forKey:(NSString *)key;
- (NSInteger)integerForKey:(NSString *)key;

    // NSInteger is a typedef to 32 or 64 bit int

- (void)setObject:(id)obj forKey:(NSString *)key;
    // obj must be a Property List
- (NSArray *)arrayForKey:(NSString *)key;
    // will return nil if value for key is not NSArray
```

```
[[NSUserDefaults standardUserDefaults]
synchronize];
```

Summary

- Foundation Data Types
- Primitive Object
- Collection
- Mutability, Enumeration, and Sorting
- Sneak Peek at Property List and User Defaults