FoundationFramework

NSObject

- (NSString *)description

Provides us with a string description of the object

NSString

- (NSString *)stringByAppendingString:(NSString *)string

Creates a new string by adding a new string to the end of it. This does not change the original string - instead it returns a new string.

- (NSString *)stringWithFormat:(NSString *)string

Creates a new string allowing us to add variables into the string. Such as allowing us to add an objects description, an int, a float, etc. into the middle of a string. We use this if we want to add variables to a string

NSData

- Used to save/restore/transmit data throughout the sdk

NSDate

- Used to create any date
- To get the current date use the ‘date’ instance method
  - [NSDate date]

NSArray

+ (id)arrayWithObjects:(id)firstObject...,nil

Creates a new array with specified objects, this must be nil terminated

+ (id)arrayWithObject:(id)soleObject

Creates a new array with only one object

- (int)count

Returns the number of objects in the array

- (id)objectAtIndex:(int)index

Returns the object at ‘index’

- (id)lastObject

Returns the last object in the array

NSArray

- (NSArray *)sortedArrayUsingSelector:(SEL)aSelector

    Sorts an array using a selector (this is just a pointer to a method)

- (NSString *)componentsJoinedByString:(NSString *)separator

    Joins all of the objects in the array together into one long string, with separator in between each object

- (BOOL)containsObject:(id)anObject

    Returns whether or not the array has anObject within it

NSMutableArray

Everything that NSArray has - plus some

NSMutableArray is a subclass of NSArray, so anything that can be done with NSArray can be done with an NSMutableArray

+ (id)array

Creates an NSMutableArray, this is a shortcut for typing [[NSMutableArray alloc] init] instead we can type [NSMutableArray array]

- (void)addObject:(id)anObject

Adds anObject to the array

- (void)insertObject:(id)anObject atIndex:(int)index

Adds anObject to the array at a specific index, everything that was at that index or later is "bumped" back by one index

- (void)removeObjectAtIndex:(int)index

Remove the object at a specific index

- (void)removeLastObject

Removes the last object in the NSArray

NSDictionary

+ (id)dictionaryWithObjects:(NSArray *)values forKeys:(NSArray *)keys

Creates a dictionary object with a set of values and keys in array form. Each value at a specific index is tied to the corresponding key at the same index.

+ (id)dictionaryWithObjectsAndKeys:(id)firstObject,firstKey...nil

Creates a dictionary by listing each object followed by the key for that object. This list must be nil terminated.

- (int)count

Returns how many objects are in the dictionary

- (id)objectForKey:(id)key

Returns the object for a specific key

- (NSArray *)allKeys

Returns an NSArray of all available keys

- (NSArray)allValues

Returns an NSArray of all available values

NSMutableDictionary

+ (id)dictionary

Returns a newly instantiated NSMutableDictionary. This is just a shortcut for [[NSMutableDictionary alloc] init], instead we can type [NSMutableDictionary dictionary]

- (void)setObject:(id)anObject forKey:(id)key

Adds a specific object/key pair to the dictionary

- (void)removeObjectForKey:(id)key

Removes the object that matches for that key from the dictionary

- (void)removeAllObjects

Removes all objects from the dictionary

- (void)addEntriesFromDictionary:(NSDictionary *)otherDictionary

Appends another dictionaries contents onto the current dictionary

NSSet

+ (id)setWithObjects:(id)firstObject,...,nil

Creates a set with the specied objects

+ (id)setWithArray:(NSArray *)anArray

Returns a set created from an array

- (int)count

Returns the number of objects in the set

- (BOOL)containsObject:(id)anObject

Returns true if the set contains an object

- (id)anyObject

Returns a psuedo-random object from the set

- (void)makeObjectsPerformSelector:(SEL)aSelector

Makes each object in the set perform a specific selector (action/method).

NSMutableSet

- (void)addObject:(id)anObject

  Adds an object to the set

- (void)removeObject:(id)anObject

  Removes an object from the set

- (void)unionSet:(NSSet *)otherSet

  Unions two sets together, this will return a set which contains objects that were in either set

- (void)minusSet:(NSSet *)otherSet

  Returns a set that includes elements that were in only one of the sets

- (void)intersectSet:(NSSet *)otherSet

  Intersects two sets together, this will return a set which contains objects that were in both sets

- (void)makeObjectsPerformSelector:(SEL)aSelector

  Makes each object in the set perform a specific selector (action/method).

NSOrderedSet

- Combination NSArray/NSSet
- Also a mutable version, NSMutableOrderedSet

Enumeration

We could do it this way

```
NSArray *myArray = ...;
for (int i=0; i < myArray.count; i++) {
        NSString *string = [myArray objectAtIndex:i];
}
```

Or We could do it this way

```
NSArray *myArray = ...;
for (NSString *string in myArray) {
        // Object is equal to string
}
```

Property List's

- A Property List is any data object that is composed of only nested versions of NSArray, NSDictionary, NSNumber, NSString, NSDate, NSData

- We can write out and read these property list files to disk to save key data on the iOS device for persistent storage between uses

NSUserDefaults

- Think of it as a storage mechanism for very small property lists
- Basically it is an NSDictionary that is persistent between launches of your application