**Name: Mahaboobsab Nadaf**

Objective C NSArray

* Data storage and its retrieval is one the most important in any program.
* NSArray is Objective-C’s general-purpose array type.
* **NSArray** and its subclass [NSMutableArray](https://developer.apple.com/library/mac/documentation/Cocoa/Reference/Foundation/Classes/NSMutableArray_Class/index.html#//apple_ref/occ/cl/NSMutableArray) manage ordered collections of objects called **arrays**.
* NSArray creates static arrays, and NSMutableArray creates dynamic arrays. You can use arrays when you need an ordered collection of objects
* The NSArray class is used for managing an ordered collection of objects.
* An ordered collection of objects is a grouping of objects that is expected to be maintained in the order in which they were stored.
* Typically, an ordered collection of objects is accessed either through enumeration or by index.
* NSArray is used to hold an immutable array of objects.
* Mutability helps to change the array in runtime a pre allocated array.
* If we use NSArray, we only replace the existing array and cannot change the contents of the existing array.
* NSArray is immutable, so you cannot dynamically add or remove items.
* The NSArray class is immutable — once it is created, you cannot modify its contents.

However, because Objective-C provides no mechanism for ensuring the immutability of the objects inside an array, if you access an element of an array, those objects can be modified.

* NSArray as well as all collections in Objective-C is zero based. This means the first element starts at index 0, and the last element has an index of one less than the length of the array.
* If you try to access an element outside of these index boundaries, you will get an exception.

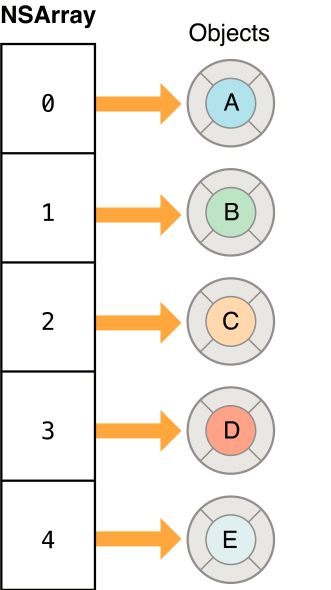


Fig a): NSArray collection classes of the Foundation Framework.

Courtesy: Apple Documentation

**Creating Arrays:**

Immutable arrays can be defined as literals using the @[] syntax.

**Example:**

NSArray \*germanMakes = @[@"Mercedes-Benz", @"BMW", @"Porsche",@"Opel", @"Volkswagen", @"Audi"];

NSArray \*ukMakes = [NSArray arrayWithObjects:@"Aston Martin",@"Lotus", @"Jaguar", @"Bentley", nil];

NSLog(@"First german make: %@", germanMakes[0]);

NSLog(@"First U.K. make: %@", [ukMakes objectAtIndex:0]);

## Enumerating Arrays:

Fast-enumeration is the most efficient way to iterate over an NSArray, and its contents are guaranteed to appear in the correct order. It’s also possible to use the count method with a traditional for-loop to step through each element in the array:

**Example:**

NSArray \*germanMakes = @[@"Mercedes-Benz", @"BMW", @"Porsche", @"Opel", @"Volkswagen", @"Audi"];

**// With fast-enumeration**

for (NSString \*item in germanMakes) {

NSLog(@"%@", item);

}

**// With a traditional for loop**

for (int i=0; i<[germanMakes count]; i++) {

NSLog(@"%d: %@", i, germanMakes[i]);

}

**Important methods of NSArray are as follows**

* alloc/initWithObjects: Used to initialize an array with objects.
* objectAtIndex: Returns the object at specific index.
* count: Returns the number of objects
* lastObject: which returns the last element of the array. To find the index of a specific element, you can also use the method
* isEqual: message to each of the elements of the array, and returns the first element which returns YES.

## Comparing Arrays:

Arrays can be compared for equality with the aptly namedisEqualToArray: method, which returns YES when both arrays have the same number of elements and every pair pass an isEqual:comparison. NSArray does not offer the same subset and intersection comparisons as NSSet.

**Example:**

NSArray \*germanMakes = @[@"Mercedes-Benz", @"BMW", @"Porsche", @"Opel", @"Volkswagen", @"Audi"];

NSArray \*sameGermanMakes = [NSArray arrayWithObjects:@"Mercedes-Benz" @"BMW", @"Porsche", @"Opel” @"Volkswagen", @"Audi", nil];

if ([germanMakes isEqualToArray:sameGermanMakes]) {

NSLog(@"Oh good, literal arrays are the same as NSArrays");

}

**Sorting Arrays**

Sorting is one of the main advantages of arrays. One of the most flexible ways to sort an array is with thesortedArrayUsingComparator: method. This accepts an^NSComparisonResult(id obj1, id obj2) block, which should return one of the following enumerators depending on the relationship between obj1 and obj2:

| **Return Value** | **Description** |
| --- | --- |
| NSOrderedAscending | obj1 comes before obj2 |
| NSOrderedSame | obj1 and obj2 have no order |
| NSOrderedDescending | obj1 comes after obj2 |

The following example sorts a list of car manufacturers based on how long their name is, from shortest to longest.

**Example:**

NSArray \*germanMakes = @[@"Mercedes-Benz", @"BMW", @"Porsche",

@"Opel", @"Volkswagen", @"Audi"];

NSArray \*sortedMakes = [germanMakes sortedArrayUsingComparator:

^NSComparisonResult(id obj1, id obj2) {

if ([obj1 length] < [obj2 length]) {

return NSOrderedAscending;

} else if ([obj1 length] > [obj2 length]) {

return NSOrderedDescending;

} else {

return NSOrderedSame;

}

}];

NSLog(@"%@", sortedMakes);

Like NSSet, NSArray is immutable, so the sorted array is actually a new array, though it still references the same elements as the original array (this is the same behavior as NSSet).

## Filtering Arrays

You can filter an array with the filteredArrayUsingPredicate:method. A short introduction to predicates can be found in the [NSSet](http://rypress.com/tutorials/objective-c/data-types/nsset.html" \l "filtering-with-predicates)module, and a minimal example is included below. Just as with the sort method discussed above, this generates a brand new array.

**Example:**

NSArray \*germanMakes = @[@"Mercedes-Benz", @"BMW", @"Porsche",

@"Opel", @"Volkswagen", @"Audi"];

NSPredicate \*beforeL = [NSPredicate predicateWithBlock:

^BOOL(id evaluatedObject, NSDictionary \*bindings) {

NSComparisonResult result = [@"L" compare:evaluatedObject];

if (result == NSOrderedDescending) {

return YES;

} else {

return NO;

}

}];

NSArray \*makesBeforeL = [germanMakes

filteredArrayUsingPredicate:beforeL];

NSLog(@"%@", makesBeforeL); // BMW, Audi

## Subdividing Arrays

Subdividing an array is essentially the same as extracting substrings from an NSString, but instead of substringWithRange:, you usesubarrayWithRange:, as shown below.

**Example:**

NSArray \*germanMakes = @[@"Mercedes-Benz", @"BMW", @"Porsche",

@"Opel", @"Volkswagen", @"Audi"];

NSArray \*lastTwo = [germanMakes subarrayWithRange:NSMakeRange(4, 2)];

NSLog(@"%@", lastTwo); // Volkswagen, Audi

## Combining Arrays

Arrays can be combined via arrayByAddingObjectsFromArray:. As with all of the other immutable methods discussed above, these returns anew array containing all of the elements in the original array, along with the contents of the parameter.

**Example:**

NSArray \*germanMakes = @[@"Mercedes-Benz", @"BMW", @"Porsche",

@"Opel", @"Volkswagen", @"Audi"];

NSArray \*ukMakes = @[@"Aston Martin", @"Lotus", @"Jaguar", @"Bentley"];

NSArray \*allMakes = [germanMakes arrayByAddingObjectsFromArray:ukMakes];

NSLog(@"%@", allMakes);

## String Conversion

The componentsJoinedByString: method concatenates each element of the array into a string, separating them by the specified symbol(s).

**Example:**

NSArray \*ukMakes = @[@"Aston Martin", @"Lotus", @"Jaguar", @"Bentley"];

NSLog(@"%@", [ukMakes componentsJoinedByString:@", "]);

This can be useful for regular expression generation, file path manipulation, and rudimentary CSV processing; however, if you’re doing serious work with file paths/data, you’ll probably want to look for a dedicated library.

**Inherits From:**

[NSObject](https://developer.apple.com/library/mac/documentation/Cocoa/Reference/Foundation/Classes/NSObject_Class/index.html#//apple_ref/occ/cl/NSObject)

NSArray

[NSMutableArray](https://developer.apple.com/library/mac/documentation/Cocoa/Reference/Foundation/Classes/NSMutableArray_Class/index.html#//apple_ref/occ/cl/NSMutableArray)

**Import Statement:**

* @import Foundation;

**Availability:**

* Available in OS X v10.0 and later

**Advantages:**

* It represents an ordered collection of objects, and it provides a high-level interface for sorting and otherwise manipulating lists of data.

**Disadvantage:**

* The NSArray class is immutable — once it is created, you cannot modify its contents.

**References:**

* <https://developer.apple.com/library/mac/documentation/Cocoa/Reference/Foundation/Classes/NSArray_Class/index.html#//apple_ref/occ/cl/NSArray>
* <http://rypress.com/tutorials/objective-c/data-types/nsarray>
* <http://as.wiley.com/WileyCDA/Section/id-400181.html>
* <http://www.tutorialspoint.com/objective_c/objective_c_data_storage.htm>

Objective C NSMutableArray

* NSMutableArray is Objective-C’s general-purpose array type.
* NSMutableArray is subclass of NSArray.
* NSMutableArray is inherited from NSArray and hence all instance methods of NSArray is available in NSMutableArray.
* NSMutableArray creates dynamic arrays.
* The NSMutableArray class lets you dynamically add or remove items from arbitrary locations in the collection.
* Mutability helps to change the array in runtime a pre allocated array.
* If we use NSMutableArray, we can change the contents of the existing array.
* NSMutableArray are often used to represent the state of a system, but the fact that NSMutableArray records the order of its elements opens up new modeling opportunities.
* NSMutableArray can record the order in which they should be fixed.

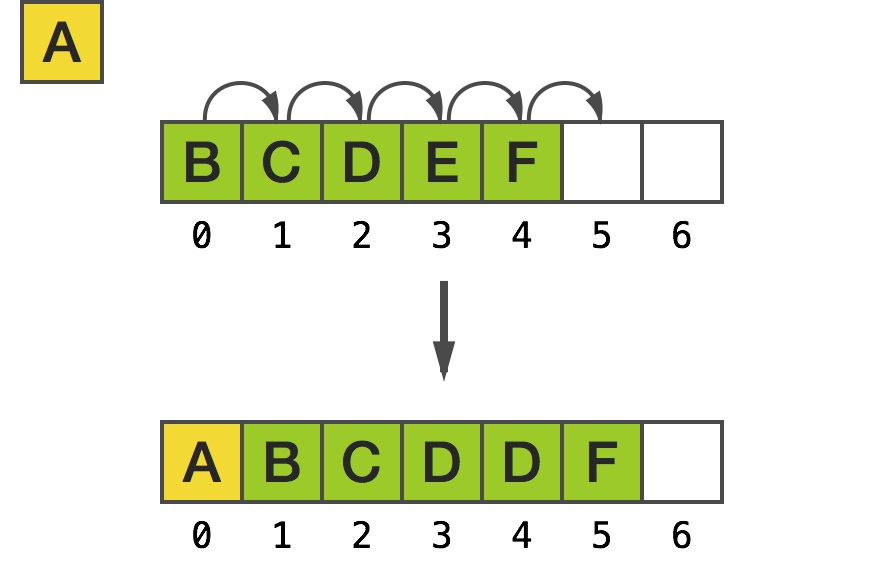


Fig a): NSMutableArray  collection classes of the Foundation Framework.

**Important methods of NSMutableArray are as follows:**

* removeAllObjects: Empties the array.
* addObject: Inserts a given object at the end of the array.
* removeObjectAtIndex: This is used to remove objectAt a specific index
* exchangeObjectAtIndex:withObjectAtIndex: Exchanges the objects in the array at given indices.
* replaceObjectAtIndex:withObject: Replaces the object at index with an Object.

## Creating Mutable Arrays:

**Example:**

NSMutableArray \*brokenCars = [NSMutableArray arrayWithObjects:@"Audi A6", @"BMW Z3", @"Audi Quattro", @"Audi TT", nil];

## Adding and Removing Objects:

The two basic methods for manipulating the contents of an array are the addObject: and removeLastObject methods. The former adds an object to the end of the array, and the latter is pretty self-documenting. Note that these are also useful methods for treating an NSArray as a stack.

**Example:**

NSMutableArray \*brokenCars = [NSMutableArray arrayWithObjects: @"Audi A6", @"BMW Z3", @"Audi Quattro", @"Audi TT", nil];

[brokenCars addObject:@"BMW F25"];

NSLog(@"%@", brokenCars); // BMW F25 added to end

[brokenCars removeLastObject];

NSLog(@"%@", brokenCars); // BMW F25 removed from end

**Inherits From:**

[NSObject](https://developer.apple.com/library/mac/documentation/Cocoa/Reference/Foundation/Classes/NSObject_Class/index.html#//apple_ref/occ/cl/NSObject)

NSArray

[NSMutableArray](https://developer.apple.com/library/mac/documentation/Cocoa/Reference/Foundation/Classes/NSMutableArray_Class/index.html#//apple_ref/occ/cl/NSMutableArray)

**Import Statement:**

* @import Foundation;

**Availability:**

* Available in OS X v10.0 and later

**Advantages of NSMutableArray over NSArray:**

* Changing how NSMutableArray stores the elements of its collection. You might do this for performance reasons or for better compatibility with legacy code.
* Acquiring more information about what is happening to the collection (for example, statistics gathering).

**Disadvantages:**

* In a subclass, you must override all the methods. You must also override the primitive methods of the [NSArray](https://developer.apple.com/library/mac/documentation/Cocoa/Reference/Foundation/Classes/NSArray_Class/index.html#//apple_ref/occ/cl/NSArray) class.

**References:**

* <https://developer.apple.com/library/mac/documentation/Cocoa/Reference/Foundation/Classes/NSArray_Class/index.html#//apple_ref/occ/cl/NSArray>
* <http://rypress.com/tutorials/objective-c/data-types/nsarray>
* <http://as.wiley.com/WileyCDA/Section/id-400181.html>
* <http://www.tutorialspoint.com/objective_c/objective_c_data_storage.htm>