Name: Mahaboobsab Nadaf

NSDictionary

* The NSDictionary class declares the programmatic interface to objects that manage immutable associations of keys and values.
* NSDictionary is used to hold an immutable dictionary of objects.
* NSDictionary is immutable.
* NSDictionary creates static dictionaries.
* The NSDictionary class represents an unordered collection of objects.
* The term **dictionary** refers to any instance of one of these classes without specifying its exact class membership.
* A key-value pair within a dictionary is called an entry.
* Each entry consists of one object that represents the key and a second object that is that key’s value.
* Within a dictionary, the keys are unique.
* No two keys in a single dictionary are equal (as determined by [isEqual:](https://developer.apple.com/library/mac/documentation/Cocoa/Reference/Foundation/Protocols/NSObject_Protocol/index.html#//apple_ref/occ/intfm/NSObject/isEqual:)).
* In general, a key can be any object, but note that when using key-value coding the key must be a string.
* Neither a key nor a value can be nil; if you need to represent a null value in a dictionary, you should use [NSNull](https://developer.apple.com/library/mac/documentation/Cocoa/Reference/Foundation/Classes/NSNull_Class/index.html#//apple_ref/occ/cl/NSNull).

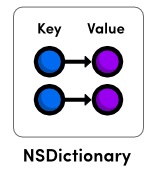


Fig a): NSDictionary collection classes of the Foundation Framework.

Courtesy: <http://rypress.com/>

**Important methods of NSDictionary are as follows**

* alloc/initWithObjectsAndKeys: Initializes a newly allocated dictionary with entries constructed from the specified set of values and keys.
* valueForKey: Returns the value associated with a given key.
* count: Returns the number of entries in the dictionary.

## Creating Dictionaries

Immutable dictionaries can be defined using the literal @{} syntax.

**Example:**

// Literal syntax

NSDictionary \*inventory = @{ @"Mercedes-Benz SLK250" : [NSNumber numberWithInt:13],@"Mercedes-Benz E350" : [NSNumber numberWithInt:22],@"BMW M3 Coupe" : [NSNumber numberWithInt:19], @"BMW X6" : [NSNumber numberWithInt:16],

};

// Values and keys as arguments

inventory = [NSDictionary dictionaryWithObjectsAndKeys:

[NSNumber numberWithInt:13], @"Mercedes-Benz SLK250", [NSNumber numberWithInt:22], @"Mercedes-Benz E350", [NSNumber numberWithInt:19], @"BMW M3 Coupe", [NSNumber numberWithInt:16], @"BMW X6", nil];

// Values and keys as arrays

NSArray \*models = @[@"Mercedes-Benz SLK250", @"Mercedes-Benz E350",@"BMW M3 Coupe", @"BMW X6"];

NSArray \*stock = @[[NSNumber numberWithInt:13],

[NSNumber numberWithInt:22]

[NSNumber numberWithInt:19],

[NSNumber numberWithInt:16]];

inventory = [NSDictionary dictionaryWithObjects:stock forKeys:models];

NSLog(@"%@", inventory);

## Accessing Values and Keys:

You can use the same subscripting syntax as arrays (someDict[key]) to access the value for a particular key. The objectForKey: method is the other common way to access values.

NSDictionary \*inventory = @{

@"Mercedes-Benz SLK250" : [NSNumber numberWithInt:13],

@"Mercedes-Benz E350" : [NSNumber numberWithInt:22],

@"BMW M3 Coupe" : [NSNumber numberWithInt:19],

@"BMW X6" : [NSNumber numberWithInt:16],

};

NSLog(@"There are %@ X6's in stock", inventory[@"BMW X6"]);

NSLog(@"There are %@ E350's in stock",

[inventory objectForKey:@"Mercedes-Benz E350"]);

## Enumerating Dictionaries:

As with sets and arrays, fast-enumeration is the most efficient way to enumerate a dictionary, and it loops through the keys (not the values).NSDictionary also defines a count method, which returns the number of entries in the collection.

NSDictionary \*inventory = @{

@"Mercedes-Benz SLK250" : [NSNumber numberWithInt:13],

@"Mercedes-Benz E350" : [NSNumber numberWithInt:22],

@"BMW M3 Coupe" : [NSNumber numberWithInt:19],

@"BMW X6" : [NSNumber numberWithInt:16],

};

NSLog(@"We currently have %ld models available", [inventory count]);

for (id key in inventory) {

NSLog(@"There are %@ %@'s in stock", inventory[key], key);

}

**Inherits From:**

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

NSDictionary

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

**Import Statement:**

* @import Foundation;

**Availability:**

* Available in OS X v10.0 and later

**Advantages:**

* It stores two objects one is key and another one is value.
* A dictionary is also a form of hash table.

**Disadvantage:**

* It can store neither a key nor a value can be nil.

**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>

NSMutableDictionary

* The NSMutableDictionary class declares the programmatic interface to objects that manage mutable associations of keys and values.
* It is subclass of NSDictionary.
* It adds modification operations to the basic operations it inherits from [NSDictionary](https://developer.apple.com/library/mac/documentation/Cocoa/Reference/Foundation/Classes/NSDictionary_Class/index.html#//apple_ref/occ/cl/NSDictionary).
* [NSMutableDictionary](http://rypress.com/tutorials/objective-c/data-types/nsdictionary#nsmutabledictionary) data structure lets you dynamically add and remove entries as necessary.
* The NSMutableDictionary class lets you add new key-value pairs dynamically.
* Mutable collections in general lend themselves to representing system states, and mutable dictionaries are no different.
* NSMutableDictionary is inherited from NSDictionary and hence all instance methods of NSDictionary are available in NSMutableDictionary.

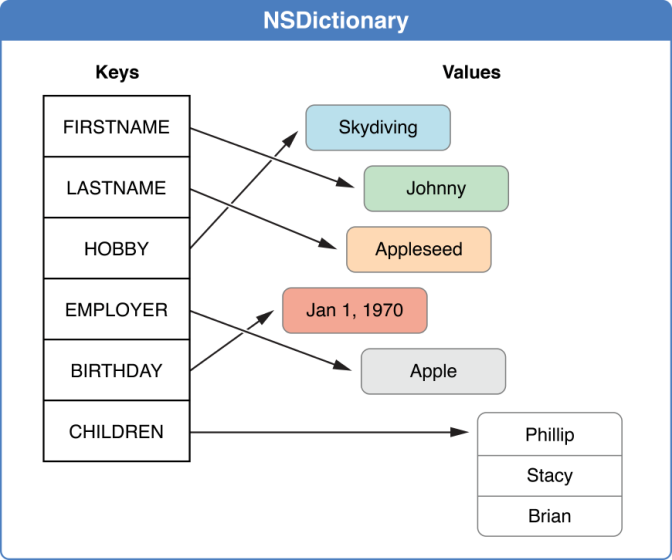


Fig a): NSDictionary collection classes of the Foundation Framework.

Courtesy: Apple Documentation

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

* removeAllObjects: Empties the dictionary of its entries.
* removeObjectForKey: Removes a given key and its associated value from the dictionary.
* setValue:forKey: Adds a given key-value pair to the dictionary.

## Creating Mutable Dictionaries:

Mutable dictionaries can be created by calling any of the factory methods defined by NSDictionary on the NSMutableDictionary class.

NSMutableDictionary \*jobs = [NSMutableDictionary

dictionaryWithDictionary:@{

@"Audi TT" : @"John",

@"Audi Quattro (Black)" : @"Mary",

@"Audi Quattro (Silver)" : @"Bill",

@"Audi A7" : @"Bill"

}];

NSLog(@"%@", jobs);

## Adding and Removing Entries:

The setObject:forKey: and removeObjectForKey: methods are the significant additions contributed by NSMutableDictionary.

NSMutableDictionary \*jobs = [NSMutableDictionary

dictionaryWithDictionary:@{

@"Audi TT" : @"John",

@"Audi Quattro (Black)" : @"Mary",

@"Audi Quattro (Silver)" : @"Bill",

@"Audi A7" : @"Bill"

}];

// Transfer an existing job to Mary

[jobs setObject:@"Mary" forKey:@"Audi TT"];

// Finish a job

[jobs removeObjectForKey:@"Audi A7"];

// Add a new job

jobs[@"Audi R8 GT"] = @"Jack";

## Enumeration Considerations:

* Dictionaries should not be mutated while you’re iterating over them.
* dictionaryWithDictionary: class method to create a shallow copy of the entire dictionary.

**Inherits From:**

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

NSDictionary

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

**Import Statement:**

* @import Foundation;

**Availability:**

* Available in OS X v10.0 and later

**Advantages:**

* It stores two objects one is key and another one is value dynamically.
* It is better choice than mutable arrays.
* A dictionary is also a form of hash table.

**Disadvantage:**

* In a subclass, you must override both of its primitive methods:
* setObject:forKey
* removeObjectForKey
* You must also override the primitive methods of the [NSDictionary](https://developer.apple.com/library/mac/documentation/Cocoa/Reference/Foundation/Classes/NSDictionary_Class/index.html#//apple_ref/occ/cl/NSDictionary) 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>