
NSDictionary Class Reference

Data Management: Data Types & Collections



2009-08-28



Apple Inc.
© 2009 Apple Inc.
All rights reserved.

No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, mechanical, electronic, photocopying, recording, or otherwise, without prior written permission of Apple Inc., with the following exceptions: Any person is hereby authorized to store documentation on a single computer for personal use only and to print copies of documentation for personal use provided that the documentation contains Apple's copyright notice.

The Apple logo is a trademark of Apple Inc.

Use of the "keyboard" Apple logo (Option-Shift-K) for commercial purposes without the prior written consent of Apple may constitute trademark infringement and unfair competition in violation of federal and state laws.

No licenses, express or implied, are granted with respect to any of the technology described in this document. Apple retains all intellectual property rights associated with the technology described in this document. This document is intended to assist application developers to develop applications only for Apple-labeled computers.

Every effort has been made to ensure that the information in this document is accurate. Apple is not responsible for typographical errors.

Apple Inc.
1 Infinite Loop
Cupertino, CA 95014
408-996-1010

Apple, the Apple logo, Cocoa, ColorSync, Mac, Mac OS, and Quartz are trademarks of Apple Inc., registered in the United States and other countries.

Spotlight is a trademark of Apple Inc.

OpenGL is a registered trademark of Silicon Graphics, Inc.

Simultaneously published in the United States and Canada.

Even though Apple has reviewed this document, APPLE MAKES NO WARRANTY OR REPRESENTATION, EITHER EXPRESS OR IMPLIED, WITH RESPECT TO THIS DOCUMENT, ITS QUALITY, ACCURACY, MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE. AS A RESULT, THIS DOCUMENT IS PROVIDED "AS IS," AND YOU, THE READER, ARE

ASSUMING THE ENTIRE RISK AS TO ITS QUALITY AND ACCURACY.

IN NO EVENT WILL APPLE BE LIABLE FOR DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES RESULTING FROM ANY DEFECT OR INACCURACY IN THIS DOCUMENT, even if advised of the possibility of such damages.

THE WARRANTY AND REMEDIES SET FORTH ABOVE ARE EXCLUSIVE AND IN LIEU OF ALL OTHERS, ORAL OR WRITTEN, EXPRESS OR IMPLIED. No Apple dealer, agent, or employee is authorized to make any modification, extension, or addition to this warranty.

Some states do not allow the exclusion or limitation of implied warranties or liability for incidental or consequential damages, so the above limitation or exclusion may not apply to you. This warranty gives you specific legal rights, and you may also have other rights which vary from state to state.

Contents

NSDictionary Class Reference 5

Overview	5
Enumeration	6
Primitive Methods	6
Descriptions and Persistence	7
Toll-Free Bridging	7
Adopted Protocols	7
Tasks	7
Creating a Dictionary	7
Initializing an NSDictionary Instance	8
Counting Entries	8
Comparing Dictionaries	8
Accessing Keys and Values	9
Enumerating Dictionaries	9
Sorting Dictionaries	9
Filtering Dictionaries	10
Storing Dictionaries	10
Accessing File Attributes	10
Creating a Description	11
Class Methods	11
dictionary	11
dictionaryWithContentsOfFile:	12
dictionaryWithContentsOfURL:	12
dictionaryWithDictionary:	13
dictionaryWithObject:forKey:	13
dictionaryWithObjects:forKeys:	14
dictionaryWithObjects:forKeys:count:	15
dictionaryWithObjectsAndKeys:	16
Instance Methods	16
allKeys	16
allKeysForObject:	17
allValues	18
count	18
description	19
descriptionInStringsFileFormat	19
descriptionWithLocale:	19
descriptionWithLocale:indent:	20
enumerateKeysAndObjectsUsingBlock:	21
enumerateKeysAndObjectsWithOptions:usingBlock:	21
fileCreationDate	22
fileExtensionHidden	22

- fileGroupOwnerAccountID 22
- fileGroupOwnerAccountName 23
- fileHFSCreatorCode 23
- fileHFSTypeCode 23
- fileIsAppendOnly 24
- fileIsImmutable 24
- fileModificationDate 24
- fileOwnerAccountID 25
- fileOwnerAccountName 25
- filePosixPermissions 26
- fileSize 26
- fileSystemFileNumber 27
- fileSystemNumber 27
- fileType 28
- getObjects:andKeys: 28
- initWithContentsOfFile: 28
- initWithContentsOfURL: 29
- initWithDictionary: 29
- initWithDictionary:copyItems: 30
- initWithObjects:forKeys: 30
- initWithObjects:forKeys:count: 31
- initWithObjectsAndKeys: 32
- isEqualToDictionary: 33
- keyEnumerator 33
- keysOfEntriesPassingTest: 34
- keysOfEntriesWithOptions:passingTest: 34
- keysSortedByValueUsingComparator: 35
- keysSortedByValueUsingSelector: 35
- keysSortedByValueWithOptions:usingComparator: 36
- objectEnumerator 36
- objectForKey: 37
- objectsForKeys:notFoundMarker: 38
- valueForKey: 38
- writeToFile:atomically: 39
- writeToURL:atomically: 39

Document Revision History 41

Index 43

NSDictionary Class Reference

Inherits from	NSObject
Conforms to	NSCoding NSCopying NSMutableCopying NSFastEnumeration NSObject (NSObject)
Framework	/System/Library/Frameworks/Foundation.framework
Availability	Available in Mac OS X v10.0 and later.
Declared in	NSDictionary.h NSFileManager.h NSKeyValueCoding.h
Companion guides	Collections Programming Topics for Cocoa Property List Programming Guide
Related sample code	From A View to A Movie From A View to A Picture People QTCoreVideo301 Quartz Composer WWDC 2005 TextEdit

Overview

The `NSDictionary` class declares the programmatic interface to objects that manage immutable associations of keys and values. Use this class or its subclass `NSMutableDictionary` when you need a convenient and efficient way to retrieve data associated with an arbitrary key. (For convenience, we use the term **dictionary** to refer 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. That is, no two keys in a single dictionary are equal (as determined by `isEqual:`). In general, a key can be any object (provided that it conforms to the `NSCopying` protocol—see below), but note that when using key-value coding the key must be a string (see [Key-Value Coding Fundamentals](#)). Neither a key nor a value can be `nil`; if you need to represent a null value in a dictionary, you should use `NSNull`.

An instance of `NSDictionary` is an immutable dictionary: you establish its entries when it's created and cannot modify them afterward. An instance of `NSMutableDictionary` is a mutable dictionary: you can add or delete entries at any time, and the object automatically allocates memory as needed. The dictionary classes adopt the `NSCopying` and `NSMutableCopying` protocols, making it convenient to convert a dictionary of one type to the other.

`NSDictionary` and `NSMutableDictionary` are part of a class cluster, so the objects you create with this interface are not actual instances of these two classes. Rather, the instances belong to one of their private subclasses. Although a dictionary's class is private, its interface is public, as declared by these abstract superclasses, `NSDictionary` and `NSMutableDictionary`.

Internally, a dictionary uses a hash table to organize its storage and to provide rapid access to a value given the corresponding key. However, the methods defined in this cluster insulate you from the complexities of working with hash tables, hashing functions, or the hashed value of keys. The methods described below take keys directly, not their hashed form.

Methods that add entries to dictionaries—whether as part of initialization (for all dictionaries) or during modification (for mutable dictionaries)—copy each key argument (keys must conform to the `NSCopying` protocol) and add the copies to the dictionary. Each corresponding value object receives a `retain` message to ensure that it won't be deallocated before the dictionary is through with it.

Enumeration

You can enumerate the contents of a dictionary by key or by value using the `NSEnumerator` object returned by [keyEnumerator](#) (page 33) and [objectEnumerator](#) (page 36) respectively. On Mac OS X v10.5 and later, `NSDictionary` supports the `NSFastEnumeration` protocol. You can use the `for...in` construct to enumerate the keys of a dictionary, as illustrated in the following example.

```
NSArray *keys = [NSArray arrayWithObjects:@"key1", @"key2", @"key3", nil];
NSArray *objects = [NSArray arrayWithObjects:@"value1", @"value2", @"value3",
nil];
NSDictionary *dictionary = [NSDictionary dictionaryWithObjects:objects
forKeys:keys];

for (id key in dictionary) {
    NSLog(@"key: %@, value: %@", key, [dictionary objectForKey:key]);
}
```

On Mac OS X v10.6 and later, `NSDictionary` supports enumeration using block objects.

Primitive Methods

Three primitive methods of `NSDictionary`—[count](#) (page 18), [objectForKey:](#) (page 37), and [keyEnumerator](#) (page 33)—provide the basis for all of the other methods in its interface. The [count](#) (page 18) method returns the number of entries in the dictionary. [objectForKey:](#) (page 37) returns the value associated with a given key. [keyEnumerator](#) (page 33) returns an object that lets you iterate through each of the keys in the dictionary. The other methods declared here operate by invoking one or more of these primitives. The non-primitive methods provide convenient ways of accessing multiple entries at once.

Descriptions and Persistence

You can use the `description...` and `writeToFile:atomically:` (page 39) methods to write a *property list representation* of a dictionary to a string or to a file, respectively. These are not intended to be used for general persistent storage of your custom data objects—see instead *Archives and Serializations Programming Guide for Cocoa*.

Toll-Free Bridging

`NSDictionary` is “toll-free bridged” with its Core Foundation counterpart, *CFDictionary Reference*. This means that the Core Foundation type is interchangeable in function or method calls with the bridged Foundation object. Therefore, in a method where you see an `NSDictionary *` parameter, you can pass in a `CFDictionaryRef`, and where you see a `CFDictionaryRef` parameter, you can pass in an `NSDictionary` instance (you cast one type to the other to suppress compiler warnings). This bridging also applies to concrete subclasses of `NSDictionary`. See *Interchangeable Data Types* for more information on toll-free bridging.

Adopted Protocols

NSCoding

- `encodeWithCoder:`
- `initWithCoder:`

NSCopying

- `copyWithZone:`

NSMutableCopying

- `mutableCopyWithZone:`

NSFastEnumeration

- `countByEnumeratingWithState:objects:count:`

Tasks

Creating a Dictionary

+ `dictionary` (page 11)

Creates and returns an empty dictionary.

+ `dictionaryWithContentsOfFile:` (page 12)

Creates and returns a dictionary using the keys and values found in a file specified by a given path.

+ `dictionaryWithContentsOfURL:` (page 12)

Creates and returns a dictionary using the keys and values found in a resource specified by a given URL.

- + [dictionaryWithDictionary:](#) (page 13)
Creates and returns a dictionary containing the keys and values from another given dictionary.
- + [dictionaryWithObject:forKey:](#) (page 13)
Creates and returns a dictionary containing a given key and value.
- + [dictionaryWithObjects:forKeys:](#) (page 14)
Creates and returns a dictionary containing entries constructed from the contents of an array of keys and an array of values.
- + [dictionaryWithObjects:forKeys:count:](#) (page 15)
Creates and returns a dictionary containing *count* objects from the *objects* array.
- + [dictionaryWithObjectsAndKeys:](#) (page 16)
Creates and returns a dictionary containing entries constructed from the specified set of values and keys.

Initializing an NSDictionary Instance

- [initWithContentsOfFile:](#) (page 28)
Initializes a newly allocated dictionary using the keys and values found in a file at a given path.
- [initWithContentsOfURL:](#) (page 29)
Initializes a newly allocated dictionary using the keys and values found at a given URL.
- [initWithDictionary:](#) (page 29)
Initializes a newly allocated dictionary by placing in it the keys and values contained in another given dictionary.
- [initWithDictionary:copyItems:](#) (page 30)
Initializes a newly allocated dictionary using the objects contained in another given dictionary.
- [initWithObjects:forKeys:](#) (page 30)
Initializes a newly allocated dictionary with entries constructed from the contents of the *objects* and *keys* arrays.
- [initWithObjects:forKeys:count:](#) (page 31)
Initializes a newly allocated dictionary with *count* entries.
- [initWithObjectsAndKeys:](#) (page 32)
Initializes a newly allocated dictionary with entries constructed from the specified set of values and keys.

Counting Entries

- [count](#) (page 18)
Returns the number of entries in the receiver.

Comparing Dictionaries

- [isEqualtoDictionary:](#) (page 33)
Returns a Boolean value that indicates whether the contents of the receiver are equal to the contents of another given dictionary.

Accessing Keys and Values

- [allKeys](#) (page 16)
Returns a new array containing the receiver's keys.
- [allKeysForObject:](#) (page 17)
Returns a new array containing the keys corresponding to all occurrences of a given object in the receiver.
- [allValues](#) (page 18)
Returns a new array containing the receiver's values.
- [getObjects:andKeys:](#) (page 28)
Returns by reference C arrays of the keys and values in the receiver.
- [objectForKey:](#) (page 37)
Returns the value associated with a given key.
- [objectsForKeys:notFoundMarker:](#) (page 38)
Returns the set of objects from the receiver that corresponds to the specified *keys* as an NSArray.
- [valueForKey:](#) (page 38)
Returns the value associated with a given key.

Enumerating Dictionaries

- [keyEnumerator](#) (page 33)
Returns an enumerator object that lets you access each key in the receiver.
- [objectEnumerator](#) (page 36)
Returns an enumerator object that lets you access each value in the receiver.
- [enumerateKeysAndObjectsUsingBlock:](#) (page 21)
Applies a given block object to the entries of the receiver.
- [enumerateKeysAndObjectsWithOptions:usingBlock:](#) (page 21)
Applies a given block object to the entries of the receiver.

Sorting Dictionaries

- [keysSortedByValueUsingSelector:](#) (page 35)
Returns an array of the receiver's keys, in the order they would be in if the receiver were sorted by its values.
- [keysSortedByValueUsingComparator:](#) (page 35)
Returns an array of the receiver's keys, in the order they would be in if the receiver were sorted by its values using a given comparator block.
- [keysSortedByValueWithOptions:usingComparator:](#) (page 36)
Returns an array of the receiver's keys, in the order they would be in if the receiver were sorted by its values using a given comparator block and a specified set of options.

Filtering Dictionaries

- [keysOfEntriesPassingTest:](#) (page 34)
Returns the set of keys whose corresponding value satisfies a constraint described by a block object.
- [keysOfEntriesWithOptions:passingTest:](#) (page 34)
Returns the set of keys whose corresponding value satisfies a constraint described by a block object.

Storing Dictionaries

- [writeToFile:atomically:](#) (page 39)
Writes a property list representation of the contents of the receiver to a given path.
- [writeToURL:atomically:](#) (page 39)
Writes a property list representation of the contents of the receiver to a given URL.

Accessing File Attributes

- [fileCreationDate](#) (page 22)
Returns the value for the `NSFileCreationDate` key.
- [fileExtensionHidden](#) (page 22)
Returns the value for the `NSFileExtensionHidden` key.
- [fileGroupOwnerAccountID](#) (page 22)
Returns the value for the `NSFileGroupOwnerAccountID` key.
- [fileGroupOwnerAccountName](#) (page 23)
Returns the value for the `NSFileGroupOwnerAccountName` key.
- [fileHFSCreatorCode](#) (page 23)
Returns the value for the `NSFileHFSCreatorCode` key.
- [fileHFSTypeCode](#) (page 23)
Returns the value for the `NSFileHFSTypeCode` key.
- [fileIsAppendOnly](#) (page 24)
Returns the value for the `NSFileAppendOnly` key.
- [fileIsImmutable](#) (page 24)
Returns the value for the `NSFileImmutable` key.
- [fileModificationDate](#) (page 24)
Returns the value for the key `NSFileModificationDate`.
- [fileOwnerAccountID](#) (page 25)
Returns the value for the `NSFileOwnerAccountID` key.
- [fileOwnerAccountName](#) (page 25)
Returns the value for the key `NSFileOwnerAccountName`.
- [filePosixPermissions](#) (page 26)
Returns the value for the key `NSFilePosixPermissions`.
- [fileSize](#) (page 26)
Returns the value for the key `NSFileSize`.

- [fileSystemFileNumber](#) (page 27)
Returns the value for the key `NSFileSystemFileNumber`.
- [fileSystemNumber](#) (page 27)
Returns the value for the key `NSFileSystemNumber`.
- [fileType](#) (page 28)
Returns the value for the key `NSFileType`.

Creating a Description

- [description](#) (page 19)
Returns a string that represents the contents of the receiver, formatted as a property list.
- [descriptionInStringsFileFormat](#) (page 19)
Returns a string that represents the contents of the receiver, formatted in `.strings` file format.
- [descriptionWithLocale:](#) (page 19)
Returns a string object that represents the contents of the receiver, formatted as a property list.
- [descriptionWithLocale:indent:](#) (page 20)
Returns a string object that represents the contents of the receiver, formatted as a property list.

Class Methods

dictionary

Creates and returns an empty dictionary.

```
+ (id)dictionary
```

Return Value

A new empty dictionary.

Discussion

This method is declared primarily for use with mutable subclasses of `NSDictionary`.

If you don't want a temporary object, you can also create an empty dictionary using `alloc...` and `init`.

Availability

Available in Mac OS X v10.0 and later.

Related Sample Code

Cocoa OpenGL

CustomAtomicStoreSubclass

FunHouse

Quartz Composer WWDC 2005 TextEdit

SimpleStickies

Declared In

`NSDictionary.h`

dictionaryWithContentsOfFile:

Creates and returns a dictionary using the keys and values found in a file specified by a given path.

```
+ (id)dictionaryWithContentsOfFile:(NSString *)path
```

Parameters

path

A full or relative pathname. The file identified by *path* must contain a string representation of a property list whose root object is a dictionary. The dictionary must contain only property list objects (instances of `NSData`, `NSDate`, `NSNumber`, `NSString`, `NSArray`, or `NSDictionary`). For more details, see *Property List Programming Guide*.

Return Value

A new dictionary that contains the dictionary at *path*, or `nil` if there is a file error or if the contents of the file are an invalid representation of a dictionary.

Availability

Available in Mac OS X v10.0 and later.

See Also

- [initWithContentsOfFile:](#) (page 28)

Related Sample Code

CapabilitiesSample

Cocoa - SGDataProc

LSMSmartCategorizer

Spotlight

SpotlightFortunes

Declared In

`NSDictionary.h`

dictionaryWithContentsOfURL:

Creates and returns a dictionary using the keys and values found in a resource specified by a given URL.

```
+ (id)dictionaryWithContentsOfURL:(NSURL *)aURL
```

Parameters

aURL

An URL that identifies a resource containing a string representation of a property list whose root object is a dictionary. The dictionary must contain only property list objects (instances of `NSData`, `NSDate`, `NSNumber`, `NSString`, `NSArray`, or `NSDictionary`). For more details, see *Property List Programming Guide*.

Return Value

A new dictionary that contains the dictionary at *aURL*, or `nil` if there is an error or if the contents of the resource are an invalid representation of a dictionary.

Availability

Available in Mac OS X v10.0 and later.

See Also

- [initWithContentsOfURL:](#) (page 29)

Declared In

NSDictionary.h

dictionaryWithDictionary:

Creates and returns a dictionary containing the keys and values from another given dictionary.

```
+ (id)dictionaryWithDictionary:(NSDictionary *)otherDictionary
```

Parameters

otherDictionary

A dictionary containing keys and values for the new dictionary.

Return Value

A new dictionary containing the keys and values found in *otherDictionary*.

Availability

Available in Mac OS X v10.0 and later.

See Also

- [initWithDictionary:](#) (page 29)

Related Sample Code

Departments and Employees

People

SimpleStickies

Declared In

NSDictionary.h

dictionaryWithObject:forKey:

Creates and returns a dictionary containing a given key and value.

```
+ (id)dictionaryWithObject:(id)anObject forKey:(id)aKey
```

Parameters

anObject

The value corresponding to *aKey*.

aKey

The key for *anObject*.

Return Value

A new dictionary containing a single object, *anObject*, for a single key, *aKey*.

Availability

Available in Mac OS X v10.0 and later.

See Also

+ [dictionaryWithObjects:forKeys:](#) (page 14)

+ [dictionaryWithObjects:forKeys:count:](#) (page 15)

+ [dictionaryWithObjectsAndKeys:](#) (page 16)

Related Sample Code

CoreRecipes

PDF Annotation Editor

QTCoreVideo301

Quartz Composer WWDC 2005 TextEdit

WhackedTV

Declared In

NSDictionary.h

dictionaryWithObjects:forKeys:

Creates and returns a dictionary containing entries constructed from the contents of an array of keys and an array of values.

```
+ (id)dictionaryWithObjects:(NSArray *)objects forKeys:(NSArray *)keys
```

Parameters

objects

An array containing the values for the new dictionary.

keys

An array containing the keys for the new dictionary. Each key is copied (using `copyWithZone:`; keys must conform to the `NSCopying` protocol), and the copy is added to the dictionary.

Return Value

A new dictionary containing entries constructed from the contents of *objects* and *keys*.

Discussion

This method steps through the *objects* and *keys* arrays, creating entries in the new dictionary as it goes. An `NSInvalidArgumentException` is raised if *objects* and *keys* don't have the same number of elements.

Availability

Available in Mac OS X v10.0 and later.

See Also

- [initWithObjects:forKeys:](#) (page 30)

+ [dictionaryWithObject:forKey:](#) (page 13)

+ [dictionaryWithObjects:forKeys:count:](#) (page 15)

+ [dictionaryWithObjectsAndKeys:](#) (page 16)

Related Sample Code

From A View to A Movie

From A View to A Picture

ImageMapExample

OpenGL Filter Basics Cocoa

TimelineToTC

Declared In

NSDictionary.h

dictionaryWithObjects:forKeys:count:Creates and returns a dictionary containing *count* objects from the *objects* array.+ (id)dictionaryWithObjects:(id *)*objects* forKeys:(id *)*keys* count:(NSUInteger)*count***Parameters***objects*

A C array of values for the new dictionary.

*keys*A C array of keys for the new dictionary. Each key is copied (using `copyWithZone:`; keys must conform to the `NSCopying` protocol), and the copy is added to the new dictionary.*count*The number of elements to use from the *keys* and *objects* arrays. *count* must not exceed the number of elements in *objects* or *keys*.**Discussion**This method steps through the *objects* and *keys* arrays, creating entries in the new dictionary as it goes. An `NSInvalidArgumentException` is raised if a key or value object is `nil`.

The following code fragment illustrates how to create a dictionary that associates the alphabetic characters with their ASCII values:

```
static const NSInteger N_ENTRIES = 26;
NSMutableDictionary *asciiDict;
NSString *keyArray[N_ENTRIES];
NSNumber *valueArray[N_ENTRIES];
NSInteger i;

for (i = 0; i < N_ENTRIES; i++) {

    char charValue = 'a' + i;
    keyArray[i] = [NSString stringWithFormat:@"%c", charValue];
    valueArray[i] = [NSNumber numberWithInt:charValue];
}

asciiDict = [NSMutableDictionary dictionaryWithObjects:(id *)valueArray
                                                    forKeys:(id *)keyArray count:N_ENTRIES];
```

Availability

Available in Mac OS X v10.0 and later.

See Also

- [initWithObjects:forKeys:count:](#) (page 31)
- + [dictionaryWithObject:forKey:](#) (page 13)
- + [dictionaryWithObjects:forKeys:](#) (page 14)
- + [dictionaryWithObjectsAndKeys:](#) (page 16)

Declared In

NSDictionary.h

dictionaryWithObjectsAndKeys:

Creates and returns a dictionary containing entries constructed from the specified set of values and keys.

```
+ (id)dictionaryWithObjectsAndKeys:(id)firstObject , ...
```

Parameters

firstObject

The first value to add to the new dictionary.

...

First the key for *firstObject*, then a null-terminated list of alternating values and keys. If any key is *nil*, an `NSInvalidArgumentException` is raised.

Discussion

This method is similar to `dictionaryWithObjects:forKeys:` (page 14), differing only in the way key-value pairs are specified.

For example:

```
NSMutableDictionary *dict = [NSMutableDictionary dictionaryWithObjectsAndKeys:
    @"value1", @"key1", @"value2", @"key2", nil];
```

Availability

Available in Mac OS X v10.0 and later.

See Also

- `initWithObjectsAndKeys:` (page 32)
- + `dictionaryWithObject:forKey:` (page 13)
- + `dictionaryWithObjects:forKeys:` (page 14)
- + `dictionaryWithObjects:forKeys:count:` (page 15)

Related Sample Code

CIAnnotation

FunHouse

IconCollection

People

Quartz Composer WWDC 2005 TextEdit

Declared In

NSMutableDictionary.h

Instance Methods

allKeys

Returns a new array containing the receiver's keys.

```
- (NSArray *)allKeys
```

Return Value

A new array containing the receiver's keys, or an empty array if the receiver has no entries.

Discussion

The order of the elements in the array is not defined.

Availability

Available in Mac OS X v10.0 and later.

See Also

- [allValues](#) (page 18)
- [allKeysForObject:](#) (page 17)
- [getObjects:andKeys:](#) (page 28)

Related Sample Code

Core Data HTML Store

CoreRecipes

FunHouse

ImageApp

SimpleStickies

Declared In

NSDictionary.h

allKeysForObject:

Returns a new array containing the keys corresponding to all occurrences of a given object in the receiver.

```
- (NSArray *)allKeysForObject:(id)anObject
```

Parameters

anObject

The value to look for in the receiver.

Return Value

A new array containing the keys corresponding to all occurrences of *anObject* in the receiver. If no object matching *anObject* is found, returns an empty array.

Discussion

Each object in the receiver is sent an `isEqual:` message to determine if it's equal to *anObject*.

Availability

Available in Mac OS X v10.0 and later.

See Also

- [allKeys](#) (page 16)
- [keyEnumerator](#) (page 33)

Related Sample Code

CoreRecipes

Declared In

NSDictionary.h

allValues

Returns a new array containing the receiver's values.

- (NSArray *)allValues

Return Value

A new array containing the receiver's values, or an empty array if the receiver has no entries.

Discussion

The order of the values in the array isn't defined.

Availability

Available in Mac OS X v10.0 and later.

See Also

- [allKeys](#) (page 16)
- [getObjects:andKeys:](#) (page 28)
- [objectEnumerator](#) (page 36)

Related Sample Code

ImageMap

ImageMapExample

People

Declared In

NSDictionary.h

count

Returns the number of entries in the receiver.

- (NSUInteger)count

Return Value

The number of entries in the receiver.

Availability

Available in Mac OS X v10.0 and later.

Related Sample Code

DynamicProperties

From A View to A Movie

From A View to A Picture

ImageApp

LSMSmartCategorizer

Declared In

NSDictionary.h

description

Returns a string that represents the contents of the receiver, formatted as a property list.

- (NSString *)description

Return Value

A string that represents the contents of the receiver, formatted as a property list.

Discussion

If each key in the receiver is an `NSString` object, the entries are listed in ascending order by key, otherwise the order in which the entries are listed is undefined. This method is intended to produce readable output for debugging purposes, not for serializing data. If you want to store dictionary data for later retrieval, see *Property List Programming Guide* and *Archives and Serializations Programming Guide for Cocoa*.

Availability

Available in Mac OS X v10.0 and later.

See Also

- [descriptionWithLocale:](#) (page 19)
- [descriptionWithLocale:indent:](#) (page 20)

Related Sample Code

Sketch-112
TextLinks

Declared In

NSDictionary.h

descriptionInStringsFileFormat

Returns a string that represents the contents of the receiver, formatted in `.strings` file format.

- (NSString *)descriptionInStringsFileFormat

Return Value

A string that represents the contents of the receiver, formatted in `.strings` file format.

Discussion

The order in which the entries are listed is undefined.

Availability

Available in Mac OS X v10.0 and later.

Declared In

NSDictionary.h

descriptionWithLocale:

Returns a string object that represents the contents of the receiver, formatted as a property list.

- (NSString *)descriptionWithLocale:(id)locale

Parameters*locale*

An object that specifies options used for formatting each of the receiver's keys and values; pass `nil` if you don't want them formatted.

Prior to Mac OS X v10.5, `locale` must be an instance of `NSDictionary`. With Mac OS X v10.5 and later, it may also be an `NSLocale` object.

Discussion

For a description of how *locale* is applied to each element in the receiver, see [descriptionWithLocale:indent:](#) (page 20).

If each key in the dictionary responds to `compare:`, the entries are listed in ascending order by key, otherwise the order in which the entries are listed is undefined.

Availability

Available in Mac OS X v10.0 and later.

See Also

- [description](#) (page 19)
- [descriptionWithLocale:indent:](#) (page 20)

Declared In

`NSDictionary.h`

descriptionWithLocale:indent:

Returns a string object that represents the contents of the receiver, formatted as a property list.

```
- (NSString *)descriptionWithLocale:(id)locale indent:(NSUInteger)level
```

Parameters*locale*

An object that specifies options used for formatting each of the receiver's keys and values; pass `nil` if you don't want them formatted.

Prior to Mac OS X v10.5, `locale` must be an instance of `NSDictionary`. With Mac OS X v10.5 and later, it may also be an `NSLocale` object.

level

Specifies a level of indent, to make the output more readable: set *level* to 0 to use four spaces to indent, or 1 to indent the output with a tab character

Return Value

A string object that represents the contents of the receiver, formatted as a property list.

Discussion

The returned `NSString` object contains the string representations of each of the receiver's entries. `descriptionWithLocale:indent:` obtains the string representation of a given key or value as follows:

- If the object is an `NSString` object, it is used as is.
- If the object responds to `descriptionWithLocale:indent:`, that method is invoked to obtain the object's string representation.
- If the object responds to `descriptionWithLocale:`, that method is invoked to obtain the object's string representation.

- If none of the above conditions is met, the object's string representation is obtained by invoking its `description` method.

If each key in the dictionary responds to `compare:`, the entries are listed in ascending order, by key. Otherwise, the order in which the entries are listed is undefined.

Availability

Available in Mac OS X v10.0 and later.

See Also

- [description](#) (page 19)
- [descriptionWithLocale:](#) (page 19)

Declared In

NSDictionary.h

enumerateKeysAndObjectsUsingBlock:

Applies a given block object to the entries of the receiver.

```
- (void)enumerateKeysAndObjectsUsingBlock:(void (^)(id key, id obj, BOOL *stop))block
```

Parameters

block

A block object to operate on entries in the receiver.

Discussion

If the block sets **stop* to YES, the enumeration stops.

Availability

Available in Mac OS X v10.6 and later.

See Also

- [enumerateKeysAndObjectsWithOptions:usingBlock:](#) (page 21)

Declared In

NSDictionary.h

enumerateKeysAndObjectsWithOptions:usingBlock:

Applies a given block object to the entries of the receiver.

```
- (void)enumerateKeysAndObjectsWithOptions:(NSEnumerationOptions)opts
    usingBlock:(void (^)(id key, id obj, BOOL *stop))block
```

Parameters

opts

Enumeration options.

block

A block object to operate on entries in the receiver.

Discussion

If the block sets **stop* to YES, the enumeration stops.

Availability

Available in Mac OS X v10.6 and later.

See Also

- [enumerateKeysAndObjectsUsingBlock](#): (page 21)

Declared In

NSDictionary.h

fileCreationDate

Returns the value for the `NSFileCreationDate` key.

- (NSDate *)fileCreationDate

Return Value

The value for the `NSFileCreationDate` key, or `nil` if the receiver doesn't have an entry for the key.

Availability

Available in Mac OS X v10.2 and later.

Declared In

NSFileManager.h

fileExtensionHidden

Returns the value for the `NSFileExtensionHidden` key.

- (BOOL)fileExtensionHidden

Return Value

The value for the `NSFileExtensionHidden` key, or `NO` if the receiver doesn't have an entry for the key.

Availability

Available in Mac OS X v10.1 and later.

Declared In

NSFileManager.h

fileGroupOwnerAccountID

Returns the value for the `NSFileGroupOwnerAccountID` key.

- (NSNumber *)fileGroupOwnerAccountID

Return Value

The value for the `NSFileGroupOwnerAccountID` key, or `nil` if the receiver doesn't have an entry for the key.

Availability

Available in Mac OS X v10.2 and later.

Declared In

NSFileManager.h

fileGroupOwnerAccountNameReturns the value for the `fileGroupOwnerAccountName` key.

- (NSString *)fileGroupOwnerAccountName

Return ValueThe value for the key `fileGroupOwnerAccountName`, or `nil` if the receiver doesn't have an entry for the key.**Discussion**

This and the other `file...` methods are for use with a dictionary, such as those returned from the methods `fileAttributesAtPath:traverseLink:(NSFileManager)`, `directoryAttributes:(NSDirectoryEnumerator)`, and `fileAttributes:(NSDirectoryEnumerator)`, that represents the POSIX attributes of a file or directory. This method returns the name of the corresponding file's group.

Availability

Available in Mac OS X v10.0 and later.

Declared In

NSFileManager.h

fileHFSCreatorCodeReturns the value for the `fileHFSCreatorCode` key.

- (OSType)fileHFSCreatorCode

Return ValueThe value for the `fileHFSCreatorCode` key, or 0 if the receiver doesn't have an entry for the key.**Discussion**See HFS File Types for details on the `OSType` data type.**Availability**

Available in Mac OS X v10.1 and later.

Declared In

NSFileManager.h

fileHFSTypeCodeReturns the value for the `fileHFSTypeCode` key.

- (OSType)fileHFSTypeCode

Return ValueThe value for the `fileHFSTypeCode` key, or 0 if the receiver doesn't have an entry for the key.

Discussion

See HFS File Types for details on the OSType data type.

Availability

Available in Mac OS X v10.1 and later.

Declared In

NSFileManager.h

fileIsAppendOnly

Returns the value for the NSFileAppendOnly key.

- (BOOL)fileIsAppendOnly

Return Value

The value for the NSFileAppendOnly key, or NO if the receiver doesn't have an entry for the key.

Availability

Available in Mac OS X v10.2 and later.

Declared In

NSFileManager.h

fileIsImmutable

Returns the value for the NSFileImmutable key.

- (BOOL)fileIsImmutable

Return Value

The value for the NSFileImmutable key, or NO if the receiver doesn't have an entry for the key.

Discussion

This and the other file... methods are for use with a dictionary, such as those returned from the methods fileAttributesAtPath:traverseLink:(NSFileManager), directoryAttributes(NSDirectoryEnumerator), and fileAttributes(NSDirectoryEnumerator), that represents the POSIX attributes of a file or directory.

Availability

Available in Mac OS X v10.2 and later.

Related Sample Code

Quartz Composer WWDC 2005 TextEdit

Declared In

NSFileManager.h

fileModificationDate

Returns the value for the key NSFileModificationDate.

- (NSDate *)fileModificationDate

Return Value

The value for the key `NSFileModificationDate`, or `nil` if the receiver doesn't have an entry for the key.

Discussion

This and the other `file...` methods are for use with a dictionary, such as those returned from the methods `fileAttributesAtPath:traverseLink:(NSFileManager)`, `directoryAttributes(NSDirectoryEnumerator)`, and `fileAttributes(NSDirectoryEnumerator)`, that represents the POSIX attributes of a file or directory. This method returns the date that the file's data was last modified.

Availability

Available in Mac OS X v10.0 and later.

Related Sample Code

Quartz Composer WWDC 2005 TextEdit

Declared In

`NSFileManager.h`

fileOwnerAccountID

Returns the value for the `NSFileOwnerAccountID` key.

- (NSNumber *)fileOwnerAccountID

Return Value

The value for the `NSFileOwnerAccountID` key, or `nil` if the receiver doesn't have an entry for the key.

Discussion

This and the other `file...` methods are for use with a dictionary, such as those returned from the methods `fileAttributesAtPath:traverseLink:(NSFileManager)`, `directoryAttributes(NSDirectoryEnumerator)`, and `fileAttributes(NSDirectoryEnumerator)`, that represents the POSIX attributes of a file or directory. This method returns the account name of the file's owner.

Availability

Available in Mac OS X v10.2 and later.

Declared In

`NSFileManager.h`

fileOwnerAccountName

Returns the value for the key `NSFileOwnerAccountName`.

- (NSString *)fileOwnerAccountName

Return Value

The value for the key `NSFileOwnerAccountName`, or `nil` if the receiver doesn't have an entry for the key.

Discussion

This and the other `file...` methods are for use with a dictionary, such as those returned from the methods `fileAttributesAtPath:traverseLink:(NSFileManager)`, `directoryAttributes(NSDirectoryEnumerator)`, and `fileAttributes(NSDirectoryEnumerator)`, that represents the POSIX attributes of a file or directory. This method returns the account name of the file's owner.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`NSFileManager.h`

filePosixPermissions

Returns the value for the key `NSFilePosixPermissions`.

- (NSUInteger)filePosixPermissions

Return Value

The value, as an unsigned `long`, for the key `NSFilePosixPermissions`, or 0 if the receiver doesn't have an entry for the key.

Discussion

This and the other `file...` methods are for use with a dictionary, such as those returned from the methods `fileAttributesAtPath:traverseLink:(NSFileManager)`, `directoryAttributes(NSDirectoryEnumerator)`, and `fileAttributes(NSDirectoryEnumerator)`, that represents the POSIX attributes of a file or directory. This method returns the file's permissions.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`NSFileManager.h`

fileSize

Returns the value for the key `NSFileSize`.

- (unsigned long long)fileSize

Return Value

The value, as an unsigned `long long`, for the key `NSFileSize`, or 0 if the receiver doesn't have an entry for the key.

Discussion

This and the other `file...` methods are for use with a dictionary such, as those returned from the methods `fileAttributesAtPath:traverseLink:(NSFileManager)`, `directoryAttributes(NSDirectoryEnumerator)`, and `fileAttributes(NSDirectoryEnumerator)`, that represents the POSIX attributes of a file or directory. This method returns the file's size.

Special Considerations

If the file has a resource fork, the returned value does *not* include the size of the resource fork.

Availability

Available in Mac OS X v10.0 and later.

Declared In

NSFileManager.h

fileSystemFileNumber

Returns the value for the key `NSFileSystemFileNumber`.

- (NSUInteger)fileSystemFileNumber

Return Value

The value, as an unsigned long, for the key `NSFileSystemFileNumber`, or 0 if the receiver doesn't have an entry for the key

Discussion

This and the other `file...` methods are for use with a dictionary, such as those returned from the methods `fileAttributesAtPath:traverseLink:(NSFileManager)`, `directoryAttributes(NSDirectoryEnumerator)`, and `fileAttributes(NSDirectoryEnumerator)`, that represents the POSIX attributes of a file or directory. This method returns the file's inode.

Availability

Available in Mac OS X v10.0 and later.

Declared In

NSFileManager.h

fileSystemNumber

Returns the value for the key `NSFileSystemNumber`.

- (NSInteger)fileSystemNumber

Return Value

The value, as an unsigned long, for the key `NSFileSystemNumber`, or 0 if the receiver doesn't have an entry for the key

Discussion

This and the other `file...` methods are for use with a dictionary, such as those returned from the methods `fileAttributesAtPath:traverseLink:(NSFileManager)`, `directoryAttributes(NSDirectoryEnumerator)`, and `fileAttributes(NSDirectoryEnumerator)`, that represents the POSIX attributes of a file or directory. This method returns the ID of the device containing the file.

Availability

Available in Mac OS X v10.0 and later.

Declared In

NSFileManager.h

fileType

Returns the value for the key `NSFileType`.

```
- (NSString *)fileType
```

Return Value

The value for the key `NSFileType`, or `nil` if the receiver doesn't have an entry for the key.

Discussion

This and the other `file...` methods are for use with a dictionary, such as those returned from the methods `fileAttributesAtPath:traverseLink:(NSFileManager)`, `directoryAttributes(NSDirectoryEnumerator)`, and `fileAttributes(NSDirectoryEnumerator)`, that represents the POSIX attributes of a file or directory. This method returns the file's type. Possible return values are described in the "Constants" section of `NSFileManager`.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`NSFileManager.h`

getObjects:andKeys:

Returns by reference C arrays of the keys and values in the receiver.

```
- (void)getObjects:(id *)objects andKeys:(id *)keys
```

Parameters

objects

Upon return, contains a C array of the values in the receiver.

keys

Upon return, contains a C array of the keys in the receiver.

Discussion

The elements in the returned arrays are ordered such that the first element in *objects* is the value for the first key in *keys* and so on.

Availability

Available in Mac OS X v10.5 and later.

See Also

- [allKeys](#) (page 16)
- [allValues](#) (page 18)
- [objectForKey:](#) (page 37)
- [objectsForKeys:notFoundMarker:](#) (page 38)

Declared In

`NSDictionary.h`

initWithContentsOfFile:

Initializes a newly allocated dictionary using the keys and values found in a file at a given path.

```
- (id)initWithContentsOfFile:(NSString *)path
```

Parameters

path

A full or relative pathname. The file identified by *path* must contain a string representation of a property list whose root object is a dictionary. The dictionary must contain only property list objects (instances of `NSData`, `NSDate`, `NSNumber`, `NSString`, `NSArray`, or `NSDictionary`). For more details, see *Property List Programming Guide*.

Return Value

An initialized object—which might be different than the original receiver—that contains the dictionary at *path*, or `nil` if there is a file error or if the contents of the file are an invalid representation of a dictionary.

Availability

Available in Mac OS X v10.0 and later.

See Also

+ [dictionaryWithContentsOfFile:](#) (page 12)

Declared In

`NSDictionary.h`

initWithContentsOfURL:

Initializes a newly allocated dictionary using the keys and values found at a given URL.

```
- (id)initWithContentsOfURL:(NSURL *)aURL
```

Parameters

aURL

An URL that identifies a resource containing a string representation of a property list whose root object is a dictionary. The dictionary must contain only property list objects (instances of `NSData`, `NSDate`, `NSNumber`, `NSString`, `NSArray`, or `NSDictionary`). For more details, see *Property List Programming Guide*.

Return Value

An initialized object—which might be different than the original receiver—that contains the dictionary at *aURL*, or `nil` if there is an error or if the contents of the resource are an invalid representation of a dictionary.

Availability

Available in Mac OS X v10.0 and later.

See Also

+ [dictionaryWithContentsOfURL:](#) (page 12)

Declared In

`NSDictionary.h`

initWithDictionary:

Initializes a newly allocated dictionary by placing in it the keys and values contained in another given dictionary.

```
- (id)initWithDictionary:(NSDictionary *)otherDictionary
```

Parameters*otherDictionary*

A dictionary containing keys and values for the new dictionary.

Return ValueAn initialized object—which might be different than the original receiver—containing the keys and values found in *otherDictionary*.**Availability**

Available in Mac OS X v10.0 and later.

See Also[+ dictionaryWithDictionary:](#) (page 13)**Declared In**

NSDictionary.h

initWithDictionary:copyItems:

Initializes a newly allocated dictionary using the objects contained in another given dictionary.

- (id)initWithDictionary:(NSDictionary *)*otherDictionary* copyItems:(BOOL)*flag***Parameters***otherDictionary*

A dictionary containing keys and values for the new dictionary.

*flag*A flag that specifies whether values in *otherDictionary* should be copied. If YES, the members of *otherDictionary* are copied, and the copies are added to the receiver. If NO, the values of *otherDictionary* are retained by the new dictionary.**Return Value**An initialized object—which might be different than the original receiver—containing the keys and values found in *otherDictionary*.**Discussion**Note that `copyWithZone:` is used to make copies. Thus, the receiver's new member objects may be immutable, even though their counterparts in *otherDictionary* were mutable. Also, members must conform to the `NSCopying` protocol.**Availability**

Available in Mac OS X v10.0 and later.

See Also[- initWithDictionary:](#) (page 29)**Declared In**

NSDictionary.h

initWithObjects:forKeys:Initializes a newly allocated dictionary with entries constructed from the contents of the *objects* and *keys* arrays.

```
- (id)initWithObjects:(NSArray *)objects forKeys:(NSArray *)keys
```

Parameters*objects*

An array containing the values for the new dictionary.

keys

An array containing the keys for the new dictionary. Each key is copied (using `copyWithZone:`; keys must conform to the `NSCopying` protocol), and the copy is added to the new dictionary.

Discussion

This method steps through the *objects* and *keys* arrays, creating entries in the new dictionary as it goes. An `NSInvalidArgumentException` is raised if the *objects* and *keys* arrays do not have the same number of elements.

Availability

Available in Mac OS X v10.0 and later.

See Also

+ [dictionaryWithObjects:forKeys:](#) (page 14)

- [initWithObjects:forKeys:count:](#) (page 31)

- [initWithObjectsAndKeys:](#) (page 32)

Related Sample Code

OpenGL Filter Basics Cocoa

QTCoreVideo201

QTCoreVideo301

Declared In

NSDictionary.h

initWithObjects:forKeys:count:

Initializes a newly allocated dictionary with *count* entries.

```
- (id)initWithObjects:(id *)objects forKeys:(id *)keys count:(NSUInteger)count
```

Parameters*objects*

A C array of values for the new dictionary.

keys

A C array of keys for the new dictionary. Each key is copied (using `copyWithZone:`; keys must conform to the `NSCopying` protocol), and the copy is added to the new dictionary.

count

The number of elements to use from the *keys* and *objects* arrays. *count* must not exceed the number of elements in *objects* or *keys*.

Discussion

This method steps through the *objects* and *keys* arrays, creating entries in the new dictionary as it goes. An `NSInvalidArgumentException` is raised if a key or value object is `nil`.

Availability

Available in Mac OS X v10.0 and later.

See Also

- + [dictionaryWithObjects:forKeys:count:](#) (page 15)
- [initWithObjects:forKeys:](#) (page 30)
- [initWithObjectsAndKeys:](#) (page 32)

Declared In

NSDictionary.h

initWithObjectsAndKeys:

Initializes a newly allocated dictionary with entries constructed from the specified set of values and keys.

```
- (id)initWithObjectsAndKeys:(id)firstObject , ...
```

Parameters

firstObject

The first value to add to the new dictionary.

...

First the key for *firstObject*, then a null-terminated list of alternating values and keys. If any key is `nil`, an `NSInvalidArgumentException` is raised.

Discussion

This method is similar to [initWithObjects:forKeys:](#) (page 30), differing only in the way in which the key-value pairs are specified.

For example:

```
NSDictionary *dict = [[NSDictionary alloc] initWithObjectsAndKeys:
    @"value1", @"key1", @"value2", @"key2", nil];
```

Availability

Available in Mac OS X v10.0 and later.

See Also

- + [dictionaryWithObjectsAndKeys:](#) (page 16)
- [initWithObjects:forKeys:](#) (page 30)
- [initWithObjects:forKeys:count:](#) (page 31)

Related Sample Code

QTRecorder
 Quartz Composer WWDC 2005 TextEdit
 QuickLookSketch
 Sketch+Accessibility
 SpeedometerView

Declared In

NSDictionary.h

isEqualToDictionary:

Returns a Boolean value that indicates whether the contents of the receiver are equal to the contents of another given dictionary.

```
- (BOOL)isEqualToDictionary:(NSDictionary *)otherDictionary
```

Parameters

otherDictionary

The dictionary with which to compare the receiver.

Return Value

YES if the contents of *otherDictionary* are equal to the contents of the receiver, otherwise NO.

Discussion

Two dictionaries have equal contents if they each hold the same number of entries and, for a given key, the corresponding value objects in each dictionary satisfy the `isEqual:` test.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `isEqual:` (NSObject protocol)

Declared In

NSDictionary.h

keyEnumerator

Returns an enumerator object that lets you access each key in the receiver.

```
- (NSEnumerator *)keyEnumerator
```

Return Value

An enumerator object that lets you access each key in the receiver.

Discussion

The following code fragment illustrates how you might use this method.

```
NSEnumerator *enumerator = [myDictionary keyEnumerator];
id key;

while ((key = [enumerator nextObject])) {
    /* code that uses the returned key */
}
```

If you use this method with instances of mutable subclasses of `NSDictionary`, your code should not modify the entries during enumeration. If you intend to modify the entries, use the [allKeys](#) (page 16) method to create a “snapshot” of the dictionary’s keys. Then use this snapshot to traverse the entries, modifying them along the way.

Note that the [objectEnumerator](#) (page 36) method provides a convenient way to access each value in the dictionary.

Availability

Available in Mac OS X v10.0 and later.

See Also

- [allKeys](#) (page 16)
- [allKeysForObject:](#) (page 17)
- [getObjects:andKeys:](#) (page 28)
- [objectEnumerator](#) (page 36)
- [nextObject](#) (NSEnumerator)

Related Sample Code

ColorSyncDevices-Cocoa
 LSMSmartCategorizer
 QuickLookSketch

Declared In

NSDictionary.h

keysOfEntriesPassingTest:

Returns the set of keys whose corresponding value satisfies a constraint described by a block object.

```
- (NSSet *)keysOfEntriesPassingTest:(BOOL (^)(id key, id obj, BOOL *stop))predicate
```

Parameters

predicate

A block object that specifies constraints for values in the receiver.

Return Value

The set of keys whose corresponding value satisfies *predicate*.

Availability

Available in Mac OS X v10.6 and later.

See Also

- [enumerateKeysAndObjectsUsingBlock:](#) (page 21)

Declared In

NSDictionary.h

keysOfEntriesWithOptions:passingTest:

Returns the set of keys whose corresponding value satisfies a constraint described by a block object.

```
- (NSSet *)keysOfEntriesWithOptions:(NSEnumerationOptions)opts passingTest:(BOOL  

  (^)(id key, id obj, BOOL *stop))predicate
```

Parameters

opts

A bit mask of enumeration options.

predicate

A block object that specifies constraints for values in the receiver.

Return Value

The set of keys whose corresponding value satisfies *predicate*.

Availability

Available in Mac OS X v10.6 and later.

See Also

- [enumerateKeysAndObjectsWithOptions:usingBlock:](#) (page 21)

Declared In

NSDictionary.h

keysSortedByValueUsingComparator:

Returns an array of the receiver's keys, in the order they would be in if the receiver were sorted by its values using a given comparator block.

- (NSArray *)keysSortedByValueUsingComparator:(NSComparator) *cmptr*

Parameters

cmptr

A comparator block.

Return Value

An array of the receiver's keys, in the order they would be in if the receiver were sorted by its values using *cmptr*.

Availability

Available in Mac OS X v10.6 and later.

See Also

- [keysSortedByValueWithOptions:usingComparator:](#) (page 36)

Declared In

NSDictionary.h

keysSortedByValueUsingSelector:

Returns an array of the receiver's keys, in the order they would be in if the receiver were sorted by its values.

- (NSArray *)keysSortedByValueUsingSelector:(SEL) *comparator*

Parameters

comparator

A selector that specifies the method to use to compare the values in the receiver.

The *comparator* method should return `NSOrderedAscending` if the receiver is smaller than the argument, `NSOrderedDescending` if the receiver is larger than the argument, and `NSOrderedSame` if they are equal.

Return Value

An array of the receiver's keys, in the order they would be in if the receiver were sorted by its values.

Discussion

Pairs of dictionary values are compared using the comparison method specified by *comparator*; the *comparator* message is sent to one of the values and has as its single argument the other value from the dictionary.

Availability

Available in Mac OS X v10.0 and later.

See Also

- [allKeys](#) (page 16)
- `sortedArrayUsingSelector:` (NSArray)

Declared In

NSDictionary.h

keysSortedByValueWithOptions:usingComparator:

Returns an array of the receiver's keys, in the order they would be in if the receiver were sorted by its values using a given comparator block and a specified set of options.

- (NSArray *)keysSortedByValueWithOptions:(NSSortOptions)opts
usingComparator:(NSComparator)cmptr

Parameters

opts

A bitmask of sort options.

cmptr

A comparator block.

Return Value

An array of the receiver's keys, in the order they would be in if the receiver were sorted by its values using *cmptr* with the options given in *opts*.

Availability

Available in Mac OS X v10.6 and later.

See Also

- [keysSortedByValueUsingComparator:](#) (page 35)

Declared In

NSDictionary.h

objectEnumerator

Returns an enumerator object that lets you access each value in the receiver.

- (NSEnumerator *)objectEnumerator

Return Value

An enumerator object that lets you access each value in the receiver.

Discussion

The following code fragment illustrates how you might use the method.

```

NSEnumerator *enumerator = [myDictionary objectEnumerator];
id value;

while ((value = [enumerator nextObject])) {

```

```

    /* code that acts on the dictionary's values */
}

```

If you use this method with instances of mutable subclasses of `NSDictionary`, your code should not modify the entries during enumeration. If you intend to modify the entries, use the [allValues](#) (page 18) method to create a “snapshot” of the dictionary’s values. Work from this snapshot to modify the values.

Availability

Available in Mac OS X v10.0 and later.

See Also

- [keyEnumerator](#) (page 33)
- `nextObject` (`NSEnumerator`)

Related Sample Code

FunHouse

LSMSmartCategorizer

SourceView

Declared In

`NSDictionary.h`

objectForKey:

Returns the value associated with a given key.

```
- (id)objectForKey:(id)aKey
```

Parameters

aKey

The key for which to return the corresponding value.

Return Value

The value associated with *aKey*, or `nil` if no value is associated with *aKey*.

Availability

Available in Mac OS X v10.0 and later.

See Also

- [allKeys](#) (page 16)
- [allValues](#) (page 18)
- [getObjects:andKeys:](#) (page 28)

Related Sample Code

From A View to A Movie

From A View to A Picture

FunHouse

People

Quartz Composer WWDC 2005 TextEdit

Declared In

`NSDictionary.h`

objectsForKeys:notFoundMarker:

Returns the set of objects from the receiver that corresponds to the specified *keys* as an NSArray.

```
- (NSArray *)objectsForKeys:(NSArray *)keys notFoundMarker:(id)anObject
```

Parameters

keys

The keys for which to return corresponding values.

anObject

The marker object to place in the corresponding element of the returned array if an object isn't found in the receiver to correspond to a given key.

Discussion

The objects in the returned array and the *keys* array have a one-for-one correspondence, so that the *n*th object in the returned array corresponds to the *n*th key in *keys*.

Availability

Available in Mac OS X v10.0 and later.

See Also

- [allKeys](#) (page 16)
- [allValues](#) (page 18)
- [getObjects:andKeys:](#) (page 28)

Declared In

NSDictionary.h

valueForKey:

Returns the value associated with a given key.

```
- (id)valueForKey:(NSString *)key
```

Parameters

key

The key for which to return the corresponding value. Note that when using key-value coding, the key must be a string (see Key-Value Coding Fundamentals).

Return Value

The value associated with *key*.

Discussion

If *key* does not start with “@”, invokes [objectForKey:](#) (page 37). If *key* does start with “@”, strips the “@” and invokes `[super valueForKey:]` with the rest of the key.

Availability

Available in Mac OS X v10.3 and later.

See Also

- [setValue:forKey:](#) (NSMutableDictionary)
- [getObjects:andKeys:](#) (page 28)

Related Sample Code

CoreRecipes
 CustomAtomicStoreSubclass
 FunHouse
 SimpleCalendar
 SimpleStickies

Declared In

NSKeyValueCoding.h

writeToFile:atomically:

Writes a property list representation of the contents of the receiver to a given path.

```
- (BOOL)writeToFile:(NSString *)path atomically:(BOOL)flag
```

Parameters

path

The path at which to write the file.

If *path* contains a tilde (~) character, you must expand it with `stringByExpandingTildeInPath` before invoking this method.

flag

A flag that specifies whether the file should be written atomically.

If *flag* is YES, the receiver is written to an auxiliary file, and then the auxiliary file is renamed to *path*.

If *flag* is NO, the dictionary is written directly to *path*. The YES option guarantees that *path*, if it exists at all, won't be corrupted even if the system should crash during writing.

Return Value

YES if the file is written successfully, otherwise NO.

Discussion

This method recursively validates that all the contained objects are property list objects (instances of `NSData`, `NSDate`, `NSNumber`, `NSString`, `NSArray`, or `NSDictionary`) before writing out the file, and returns NO if all the objects are not property list objects, since the resultant file would not be a valid property list.

If the receiver's contents are all property list objects, the file written by this method can be used to initialize a new dictionary with the class method `dictionaryWithContentsOfFile:` (page 12) or the instance method `initWithContentsOfFile:` (page 28).

For more information about property lists, see *Property List Programming Guide*.

Availability

Available in Mac OS X v10.0 and later.

Declared In

NSDictionary.h

writeToURL:atomically:

Writes a property list representation of the contents of the receiver to a given URL.

- (BOOL)writeToURL:(NSURL *)aURL atomically:(BOOL)flag

Parameters

aURL

The URL to which to write the receiver.

flag

A flag that specifies whether the output should be written atomically.

If *flag* is YES, the receiver is written to an auxiliary location, and then the auxiliary location is renamed to *aURL*. If *flag* is NO, the dictionary is written directly to *aURL*. The YES option guarantees that *aURL*, if it exists at all, won't be corrupted even if the system should crash during writing. *flag* is ignored if *aURL* is of a type that cannot be written atomically.

Return Value

YES if the location is written successfully, otherwise NO.

Discussion

This method recursively validates that all the contained objects are property list objects (instances of NSData, NSDate, NSNumber, NSString, NSArray, or NSDictionary) before writing out the file, and returns NO if all the objects are not property list objects, since the resultant output would not be a valid property list.

If the receiver's contents are all property list objects, the location written by this method can be used to initialize a new dictionary with the class method [dictionaryWithContentsOfURL:](#) (page 12) or the instance method [initWithContentsOfURL:](#) (page 29).

For more information about property lists, see *Property List Programming Guide*.

Availability

Available in Mac OS X v10.0 and later.

Declared In

NSDictionary.h

Document Revision History

This table describes the changes to *NSDictionary Class Reference*.

Date	Notes
2009-08-28	Updated for Mac OS X v10.6. New Blocks oriented methods for enumeration.
2009-04-08	Corrected typographical errors.
2008-10-15	Corrected wording of -description method.
2008-02-08	Corrected typographical errors.
2007-02-16	Included API introduced in Mac OS X v10.5.
2006-05-23	First publication of this content as a separate document.

REVISION HISTORY

Document Revision History

Index

A

`allKeys` **instance method** 16
`allKeysForObject:` **instance method** 17
`allValues` **instance method** 18

C

`count` **instance method** 18

D

`description` **instance method** 19
`descriptionInStringsFileFormat` **instance method** 19
`descriptionWithLocale:` **instance method** 19
`descriptionWithLocale:indent:` **instance method** 20
`dictionary` **class method** 11
`dictionaryWithContentsOfFile:` **class method** 12
`dictionaryWithContentsOfURL:` **class method** 12
`dictionaryWithDictionary:` **class method** 13
`dictionaryWithObject:forKey:` **class method** 13
`dictionaryWithObjectsAndKeys:` **class method** 16
`dictionaryWithObjects:forKeys:` **class method** 14
`dictionaryWithObjects:forKeys:count:` **class method** 15

E

`enumerateKeysAndObjectsUsingBlock:` **instance method** 21
`enumerateKeysAndObjectsWithOptions:usingBlock:` **instance method** 21

F

`fileCreationDate` **instance method** 22
`fileExtensionHidden` **instance method** 22
`fileGroupOwnerAccountID` **instance method** 22
`fileGroupOwnerAccountName` **instance method** 23
`fileHFSCreatorCode` **instance method** 23
`fileHFSTypeCode` **instance method** 23
`fileIsAppendOnly` **instance method** 24
`fileIsImmutable` **instance method** 24
`fileModificationDate` **instance method** 24
`fileOwnerAccountID` **instance method** 25
`fileOwnerAccountName` **instance method** 25
`filePosixPermissions` **instance method** 26
`fileSize` **instance method** 26
`fileSystemFileNumber` **instance method** 27
`fileSystemNumber` **instance method** 27
`fileType` **instance method** 28

G

`getObjects:andKeys:` **instance method** 28

I

`initWithContentsOfFile:` **instance method** 28
`initWithContentsOfURL:` **instance method** 29
`initWithDictionary:` **instance method** 29
`initWithDictionary:copyItems:` **instance method** 30
`initWithObjectsAndKeys:` **instance method** 32
`initWithObjects:forKeys:` **instance method** 30
`initWithObjects:forKeys:count:` **instance method** 31
`isEqualToDictionary:` **instance method** 33

K

keyEnumerator **instance method** [33](#)
keysOfEntriesPassingTest: **instance method** [34](#)
keysOfEntriesWithOptions:passingTest: **instance method** [34](#)
keysSortedByValueUsingComparator: **instance method** [35](#)
keysSortedByValueUsingSelector: **instance method** [35](#)
keysSortedByValueWithOptions:usingComparator: **instance method** [36](#)

O

objectEnumerator **instance method** [36](#)
objectForKey: **instance method** [37](#)
objectsForKeys:notFoundMarker: **instance method** [38](#)

V

valueForKey: **instance method** [38](#)

W

writeToFile:atomically: **instance method** [39](#)
writeToURL:atomically: **instance method** [39](#)