NSString

* The NSString classes represent text strings and provide methods for searching, combining, and comparing strings.
* NSString is one the most commonly used classes that is used for storing strings and texts.
* The NSString class and its mutable subclass, [NSMutableString](https://developer.apple.com/library/mac/documentation/Cocoa/Reference/Foundation/Classes/NSMutableString_Class/index.html#//apple_ref/occ/cl/NSMutableString), provide an extensive set of APIs for working with strings.
* The [NSString](https://developer.apple.com/library/mac/#documentation/Cocoa/Reference/Foundation/Classes/NSString_Class/Reference/NSString.html) class is the basic tool for representing text in an Objective-C application.
* NSString is an immutable type, so you cannot change it after it’s been instantiated.
* The string in Objective-C programming language is represented using NSString and its subclass NSMutableString provides several ways for creating string objects.
* An NSString object encodes a Unicode-compliant text string, represented as a sequence of UTF–16 code units.
* NSString objects are used extensively throughout Foundation and other Cocoa frameworks, serving as the basis for all textual and linguistic functionality on the platform
* The simplest way to create a string object is to use the Objective-C @"..." construct:
* An NSString object encodes a Unicode-compliant text string, represented as a sequence of UTF–16 code units
* All lengths, character indexes, and ranges are expressed in terms of 16-bit platform-Endean values, with index values starting at 0.
* An immutable string is a text string that is defined when it is created and subsequently cannot be changed. An immutable string is implemented as an array of Unicode characters (in other words, a text string).
* To create and manage an immutable string, use the NSString class.
* The objects you create using NSString and NSMutableString are referred to as string objects
* The term C string refers to the standard char \* type.
* string object’s class is private, its interface is public, as declared by these abstract super classes, NSString and NSMutableString.
* The string classes adopt the [NSCopying](https://developer.apple.com/library/mac/documentation/Cocoa/Reference/Foundation/Protocols/NSCopying_Protocol/index.html#//apple_ref/occ/intf/NSCopying) and [NSMutableCopying](https://developer.apple.com/library/mac/documentation/Cocoa/Reference/Foundation/Protocols/NSMutableCopying_Protocol/index.html#//apple_ref/occ/intf/NSMutableCopying) protocols, making it convenient to convert a string of one type to the other .

**Objective-C supports a wide range of methods for manipulate strings:**

|  |  |
| --- | --- |
| **S.N.** | **Method & Purpose** |
| 1 | **- (NSString \*)capitalizedString;**  Returns a capitalized representation of the receiver. |
| 2 | **- (unichar)characterAtIndex:(NSUInteger)index;**  Returns the character at a given array position. |
| 3 | **- (double)doubleValue;**  Returns the floating-point value of the receiver’s text as a double. |
| 4 | **- (BOOL)hasPrefix:(NSString \*)aString;**  Returns a Boolean value that indicates whether a given string matches the beginning characters of the receiver. |
| 5 | **- (BOOL)hasSuffix:(NSString \*)aString;**  Returns a Boolean value that indicates whether a given string matches the ending characters of the receiver. |
| 6 | **- (id)initWithFormat:(NSString \*)format ...;**  Returns an NSString object initialized by using a given format string as a template into which the remaining argument values are substituted. |
| 7 | **- (NSInteger)integerValue;**  Returns the NSInteger value of the receiver’s text. |
| 8 | **- (BOOL)isEqualToString:(NSString \*)aString;**  Returns a Boolean value that indicates whether a given string is equal to the receiver using a literal Unicode-based comparison. |
| 9 | **- (NSUInteger)length;**  Returns the number of Unicode characters in the receiver. |
| 10 | **- (NSString \*)lowercaseString;**  Returns lowercased representation of the receiver. |
| 11 | **- (NSRange)rangeOfString:(NSString \*)aString;**  Finds and returns the range of the first occurrence of a given string within the receiver. |
| 12 | **- (NSString \*)stringByAppendingFormat:(NSString \*)format ...;**  Returns a string made by appending to the receiver a string constructed from a given format string and the following arguments. |
| 13 | **- (NSString \*)stringByTrimmingCharactersInSet:(NSCharacterSet \*)set;**  Returns a new string made by removing from both ends of the receiver characters contained in a given character set. |
| 14 | **- (NSString \*)substringFromIndex:(NSUInteger)anIndex;**  Returns a new string containing the characters of the receiver from the one at a given index to the end. |

## Creating Strings:

The most common way to create strings is using the literal @"Some String" syntax, but the stringWithFormat: class method is also useful for generating strings that are composed of variable values. It takes the same kind of format string as NSLog():

NSString \*make = @"Porsche";

NSString \*model = @"911";

int year = 1968;

NSString \*message = [NSString stringWithFormat:@"That's a %@ %@ from %d!",

make, model, year];

NSLog(@"%@", message);

Notice that we used the @"%@" format specifies in the NSLog() call instead of passing the string directly with NSLog(message).

## Enumerating Strings:

The two most basic NSString methods are length andcharacterAtIndex:, which return the number of characters in the string and the character at a given index, respectively

NSString \*make = @"Porsche";

for (int i=0; i<[make length]; i++) {

unichar letter = [make characterAtIndex:i];

NSLog(@"%d: %hu", i, letter);

}

## Comparing Strings:

String comparisons present the same issues as NSNumber comparisons. Instead of comparing pointers with the == operator, you should always use the isEqualToString: method for a more robust valuecomparison. The following example shows you how this works, along with the useful hasPrefix: and hasSuffix: methods for partial comparisons.

NSString \*car = @"Porsche Boxster";

if ([car isEqualToString:@"Porsche Boxster"]) {

NSLog(@"That car is a Porsche Boxster");

}

if ([car hasPrefix:@"Porsche"]) {

NSLog(@"That car is a Porsche of some sort");

}

if ([car hasSuffix:@"Carrera"]) {

// this won't execute

NSLog(@"That car is a Carrera");

}

**Inherits From:**

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

NSString

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

**Import Statement:**

* @import Foundation;

**Availability:**

* Available in OS X v10.0 and later

**Advantages:**

* It stores any type of character.
* It has many built in methods.

**Disadvantage:**

* It is immutable.

**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>