**NSString**

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. The string in Objective-C programming language is represented using NSString and its subclass NSMutableString provides several ways for creating string objects. String objects represent character strings in Cocoa frameworks.

An immutable string is a text string that is defined when it is created and subsequently cannot be changed. To create and manage an immutable string, use the NSString class. To construct and manage a string that can be changed after it has been created, use NSMutableString. Strings can be compared against one another, search them for substrings, combine them into new strings, and so on.

**Creating a String:**

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():. A string object is an array of Unicode characters. We can use the @"%@" format specifier in the NSLog() call instead of passing the string directly with NSLog(message).

**For ex,:**

#import <Foundation/Foundation.h>

int main ()

{

NSString \*company = @"NeoRays";

NSLog(@"Company name: %@\n", company );

return 0;

}

**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. length helps to find the length of given string and characterAtIndex helps to find the character at a perticular index.

**Comparing Strings:**

String comparisons present the same issues as NSNumber comparisons. Instead of comparing pointers with the == operator, The isEqualToString: method is used for a more robust value comparison. Also hasPrefix: and hasSuffix: methods can be used for partial comparisons.

**Combining Strings:**

The methods like stringByAppendingstring and stringByAppendingFormat are used to append two strings.

**Searching String:**

NSString’s search methods all return an NSRange struct, which defines a location and a length field. The location is the index of the beginning of the match, and the length is the number of characters in the match.

**Replacing Sustrings:**

stringByReplacingCharactersInRange method is used to replace a range of a string with a new string.

**Changing case:**

The NSString class also provides a few convenient methods for changing the case of a string. This can be used to normalize user-submitted values. Methods like lowercaseString, uppercaseString and capitalizedString are used for changing the cases of strings.

**String Methods:**

**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] - (float)floatValue;**

Returns the floating-point value of the receiver’s text as a float.

**5] - (BOOL)hasPrefix:(NSString \*)aString;**

Returns a Boolean value that indicates whether a given string matches the beginning characters of the receiver.

**6] - (BOOL)hasSuffix:(NSString \*)aString;**

Returns a Boolean value that indicates whether a given string matches the ending characters of the receiver

**7] - (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.

**8] - (NSInteger)integerValue;**

Returns the NSInteger value of the receiver’s text.

**9] - (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.

**10] - (NSUInteger)length;**

Returns the number of Unicode characters in the receiver.

**11] - (NSString \*)lowercaseString;**

Returns lowercased representation of the receiver.

**12] - (NSRange)rangeOfString:(NSString \*)aString;**

Finds and returns the range of the first occurrence of a given string within the receiver.

**13] - (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.

**14] - (NSString \*)stringByTrimmingCharactersInSet:(NSCharacterSet \*)set;**

Returns a new string made by removing from both ends of the receiver characters contained in a given character set.

**15] - (NSString \*)substringFromIndex:(NSUInteger)anIndex;**

Returns a new string containing the characters of the receiver from the one at a given index to the end.

**16] isEqualToString:**

 This method is used for a more robust value comparison.

**NSMutableString**

The [NSMutableString](https://developer.apple.com/library/mac/" \l "documentation/Cocoa/Reference/Foundation/Classes/NSMutableString_Class/Reference/Reference.html) class is a mutable version of NSString. Unlike immutable strings, it’s possible to alter individual characters of a mutable string without creating a new object. NSMutableString inherits from NSString, it can be manipulated like NSString.

NSMutableString \*car = [NSMutableString stringWithString:@"Porsche 911"]

The setString: method lets you assign a new value to the instance:

[car setString:@"Porsche Boxster"];

It’s possible to replace substring via the replaceCharactersInRange:withString: and delete the substring using deleteCharactersInRange: method.

**Creating and Intializing Mutable String:**

**1]** +stringWithCapacity

#### Declaration

+ (NSMutableString \*)stringWithCapacity:([NSUInteger](https://developer.apple.com/library/mac/documentation/Cocoa/Reference/Foundation/Miscellaneous/Foundation_DataTypes/index.html" \l "//apple_ref/doc/c_ref/NSUInteger))capacity

#### Parameters

Capacity : The number of characters the string is expected to initially contain.

#### Return Value

An empty NSMutableString object with initial storage for capacity characters.

2] -initWithCapacity:

- (NSMutableString \*)initWithCapacity:([NSUInteger](https://developer.apple.com/library/mac/documentation/Cocoa/Reference/Foundation/Miscellaneous/Foundation_DataTypes/index.html" \l "//apple_ref/doc/c_ref/NSUInteger))capacity

#### Parameters

Capacity : The number of characters the string is expected to initially contain.

#### Return Value

An initialized NSMutableString object with initial storage for capacity characters. The returned object might be different than the original receiver.