NSString

Inherits From: NSObject

Conforms To: NSCoding
NSCopying
NSMutableCopying
NSObject

Declared In: foundation/NSString.h
foundation/NSPathUtilities.h
foundation/NSUtilities.h

Creating Temporary Strings

+ (NSString *)**localizedStringWithFormat:**(NSString *)*format,...*

Returns a string created by using *format* as a **printf()** style format string, and the following arguments as values to be substituted into the format string. The user's default locale is used for format information.

+ (NSString *)**stringWithCString:**(const char *)*byteString*

Returns a string containing the characters in *byteString*, which must be null-terminated. *byteString* should contain characters in the default C string encoding.

+ (NSString *)**stringWithCString:**(const char *)*byteString*
length:(unsigned int)*length*

Returns a string containing characters from *byteString*. *byteString* should contain characters in the default C string encoding. *length* bytes are copied into the string, regardless of whether a null byte exists in *byteString*.

+ (NSString *)**stringWithCharacters:**(const unichar *)*chars*

length:(unsigned int)*length*
Returns a string containing *chars*. *length* characters are copied into the string, regardless of whether a null character exists in *chars*.

+ (NSString *)**stringWithFormat:**(NSString *)*format,...*
Returns a string created by using *format* as a **printf()** style format string, and the following arguments as values to be substituted into the format string.

Initializing Newly Allocated Strings

- (id)**init**
Initializes the receiver, a newly allocated NSString, to contain no characters. This is the only initialization method that a subclass of NSString should invoke.

- (id)**initWithCString:**(const char *)*byteString*
Initializes the receiver, a newly allocated NSString, by converting the one-byte characters in *byteString* into Unicode characters. *byteString* must be a null-terminated C string in the default C string encoding.

- (id)**initWithCString:**(const char *)*byteString*
length:(unsigned int)*length*
Initializes the receiver, a newly allocated NSString, by converting *length* one-byte characters in *byteString* into Unicode characters. This method doesn't stop at a null byte.

- (id)**initWithCStringNoCopy:**(char *)*byteString*
length:(unsigned int)*length*
freeWhenDone:(BOOL)*flag*
Initializes the receiver, a newly allocated NSString, by converting *length* one-byte characters in *byteString* into Unicode characters. This method doesn't stop at a null byte. The receiver becomes the owner of *byteString;* if *flag* is YES it will free the memory when it no longer needs it, but if *flag* is NO it won't.

- (id)**initWithCharacters:**(const unichar *)*chars*
length:(unsigned int)*length*
Initializes the receiver, a newly allocated NSString, by copying *length* characters from *chars*. This method doesn't stop at a null character.

- (id)**initWithCharactersNoCopy:**(unichar *)*chars*
length:(unsigned int)*length*
freeWhenDone:(BOOL)*flag*
Initializes the receiver, a newly allocated NSString, to contain *length* characters from *chars*. This method doesn't stop at a null character. The receiver becomes the owner of *chars;* if *flag* is YES the receiver will free the memory when it no longer needs them, but if *flag* is NO it won't.

- (id)**initWithContentsOfFile:**(NSString *)*path*
Initializes the receiver, a newly allocated NSString, by reading NEXTSTEP-encoded characters from the file whose name is given by *path*.

- (id)**initWithData:**(NSData *)*data*
encoding:(NSStringEncoding)*encoding*
Initializes the receiver, a newly allocated NSString, by converting the bytes in *data* into Unicode characters. *data* must be an

NSData object containing bytes in *encoding* and in the default ªplain textº format for that encoding.

- (id)**initWithFormat:**(NSString *)*format,...* — Initializes the receiver, a newly allocated NSString, by constructing a string from *format* and following string objects in the manner of **printf()**.
- (id)**initWithFormat:**(NSString *)*format* **arguments:**(va_list)*argList* — Initializes the receiver, a newly allocated NSString, by constructing a string from *format* and *argList* in the manner of **vprintf()**.
- (id)**initWithFormat:**(NSString *)*format* **locale:**(NSDictionary *)*dictionary* — Initializes the receiver, a newly allocated NSString, by constructing a string from *format* and the formatting information in the dictionary in the manner of **printf()**.
- (id)**initWithFormat:**(NSString *)*format* **locale:**(NSDictionary *)*dictionary* **arguments:**(va_list)*argList* — Initializes the receiver, a newly allocated NSString, by constructing a string from *format* and format information in *dictionary* and *argList* in the manner of **vprintf()**.
- (id)**initWithString:**(NSString *)*string* — Initializes the receiver, a newly allocated NSString, by copying the characters from *string*.

Getting a String's Length

- (unsigned int)**length** — Returns the number of characters in the receiver. This number includes the individual characters of composed character sequences.

Accessing Characters

- (unichar)**characterAtIndex:**(unsigned int)*index* — Returns the character at the array position given by *index*. This method raises an NSStringBoundsError exception if *index* lies beyond the end of the string.
- (void)**getCharacters:**(unichar *)*buffer* — Invokes **getCharacters:range:** with the provided *buffer* and the entire extent of the receiver as the range.
- (void)**getCharacters:**(unichar *)*buffer* **range:**(NSRange)*aRange* — Copies characters from *aRange* in the receiver into *buffer*, which must be large enough to contain them. This method does *not* add a null character. This method raises an NSStringBoundsError exception if any part of *aRange* lies beyond the end of the string.

Combining Strings

- (NSString *)**stringByAppendingFormat:**(NSString *)*format,...*

Returns a string made by using *format* as a **printf()** style format string, and the following arguments as values to be substituted into the format string.

- (NSString *)**stringByAppendingString:**(NSString *)*aString*

Returns a string made by appending *aString* and the receiver.

Dividing Strings into Substrings

- (NSArray *)**componentsSeparatedByString:**(NSString *)*separator*

Finds the substrings in the receiver that are delimited by *separator* and returns them as the elements of an NSArray. The strings in the array appear in the order they did in the receiver.

- (NSString *)**substringFromIndex:**(unsigned int)*index*

Returns a string object containing the characters of the receiver starting from the one at *index* to the end. This method raises an NSStringBoundsError exception if *index* lies beyond the end of the string.

- (NSString *)**substringFromRange:**(NSRange)*aRange*

Returns a string object containing the characters of the receiver which lie within *aRange*. This method raises an NSStringBoundsError exception if any part of *aRange* lies beyond the end of the string.

- (NSString *)**substringToIndex:**(unsigned int)*index*

Returns a string object containing the characters of the receiver up to, but not including, the one at *index*. This method raises an NSStringBoundsError exception if *index* lies beyond the end of the string.

Finding Ranges of Characters and Substrings

- (NSRange)**rangeOfCharacterFromSet:**(NSCharacterSet *)*aSet*

Invokes **rangeOfCharacterFromSet:options:** with no options.

- (NSRange)**rangeOfCharacterFromSet:**(NSCharacterSet *)*aSet*
  options:(unsigned int)*mask*

Invokes **rangeOfCharacterFromSet:options:range:** with *mask* and the entire extent of the receiver as the range.

- (NSRange)**rangeOfCharacterFromSet:**(NSCharacterSet *)*aSet*
  options:(unsigned int)*mask*
  range:(NSRange)*aRange*

Returns the range of the first character found from *aSet*. The search is restricted to *aRange* with *mask* options. *mask* can be any combination (using the C bitwise OR operator |) of

NSCaseInsensitiveSearch, NSLiteralSearch, and NSBackwardsSearch.

- (NSRange)**rangeOfString:**(NSString *)*string*
  Invokes **rangeOfString:options:** with no options.
- (NSRange)**rangeOfString:**(NSString *)*string*
  options:(unsigned int)*mask*
  Invokes **rangeOfString:options:range:** with *mask* options and the entire extent of the receiver as the range.
- (NSRange)**rangeOfString:**(NSString *)*aString*
  options:(unsigned int)*mask*
  range:(NSRange)*aRange*
  Returns the range giving the location and length in the receiver of *aString*. The search is restricted to *aRange* with *mask* options. *mask* can be any combination (using the C bitwise OR operator |) of NSCaseInsensitiveSearch, NSLiteralSearch, NSBackwardsSearch, and NSAnchoredSearch.

Determining Composed Character Sequences

- (NSRange)**rangeOfComposedCharacterSequenceAtIndex:**(unsigned int)*anIndex*
  Returns an NSRange giving the location and length in the receiver of the composed character sequence located at *anIndex*. This method raises an NSStringBoundsError exception if *anIndex* lies beyond the end of the string.

Identifying and Comparing Strings

- (NSComparisonResult)**caseInsensitiveCompare:**(NSString *)*aString*
  Invokes **compare:options:** with the option NSCaseInsensitiveSearch.
- (NSComparisonResult)**compare:**(NSString *)*aString*
  Invokes **compare:options:** with no options.
- (NSComparisonResult)**compare:**(NSString *)*aString*
  options:(unsigned int)*mask*
  Invokes **compare:options:range:** with *mask* as the options and the receiver's full extent as the range.
- (NSComparisonResult)**compare:**(NSString *)*aString*
  options:(unsigned int)*mask*
  range:(NSRange)*aRange*
  Compares *aString* to the receiver and returns their lexical ordering. The comparison is restricted to *aRange* and uses *mask* options, which may be NSCaseInsensitiveSearch and NSLiteralSearch.
- (BOOL)**hasPrefix:**(NSString *)*aString*
  Returns YES if *aString* matches the beginning characters of the receiver, NO otherwise.
- (BOOL)**hasSuffix:**(NSString *)*aString*
  Returns YES if *aString* matches the ending characters of the receiver, NO otherwise.

- (unsigned int)**hash**
  Returns an unsigned integer that can be used as a table address in a hash table structure. If two string objects are equal (as determined by the **isEqual:** method), they must have the same hash value.
- (BOOL)**isEqual:**(id)*anObject*
  Returns YES if both the receiver and *anObject* have the same **id** or if they're both NSStrings that compare as **NSOrderedSame**, NO otherwise.
- (BOOL)**isEqualToString:**(NSString *)*aString*
  Returns YES if *aString* is equivalent to the receiver (if they have the same **id** or if they compare as **NSOrderedSame**), NO otherwise.

Storing the String

- (NSString *)**description**
  Returns the string itself.
- (BOOL)**writeToFile:**(NSString *)*filename*
  atomically:(BOOL)*useAuxiliaryFile*
  Writes a textual description of the receiver to *filename.*
  If *useAuxiliaryFile* is YES, the data is written to a backup file and then, assuming no errors occur, the backup file is renamed to the intended file name.

Getting a Shared Prefix

- (NSString *)**commonPrefixWithString:**(NSString *)*aString*
  options:(unsigned int)*mask*
  Returns the substring of the receiver containing characters that the receiver and *aString* have in common. *mask* can be any combination (using the C bitwise OR operator |) of NSCaseInsensitiveSearch and NSLiteralSearch.

Changing Case

- (NSString *)**capitalizedString**
  Returns a string with the first character of each word changed to its corresponding uppercase value.
- (NSString *)**lowercaseString**
  Returns a string with each character changed to its corresponding lowercase value.
- (NSString *)**uppercaseString**
  Returns a string with each character changed to its corresponding uppercase value.

Getting C Strings

- (const char *)**cString**
Returns a representation of the receiver as a C string in the default C string encoding.

- (unsigned int)**cStringLength**
Returns the length in bytes of the C string representation of the receiver.

- (void)**getCString:**(char *)*buffer*
Invokes **getCString:maxLength:range:remainingRange:** with NSMaximumStringLength as the maximum length, the receiver's entire extent as the range, and NULL for the remaining range. *buffer* must be large enough to contain the resulting C string plus a terminating null character (which this method adds).

- (void)**getCString:**(char *)*buffer*
maxLength:(unsigned int)*maxLength*
Invokes **getCString:maxLength:range:remainingRange:** with *maxLength* as the maximum length, the receiver's entire extent as the range, and NULL for the remaining range. *buffer* must be large enough to contain the resulting C string plus a terminating null character (which this method adds).

- (void)**getCString:**(char *)*buffer*
maxLength:(unsigned int)*maxLength*
range:(NSRange)*aRange*
remainingRange:(NSRange *)*leftoverRange*
Copies the receiver's characters (in the default C string encoding) as bytes into *buffer*. *buffer* must be large enough to contain *maxLength* bytes plus a terminating null character (which this method adds). Characters are copied from *aRange*; if not all characters can be copied, the range of those not copied is put into *leftoverRange*. This method raises an NSStringBoundsError exception if any part of *aRange* lies beyond the end of the string.

Getting Numeric Values

- (double)**doubleValue**
Returns the double precision floating point value of the receiver's text. Whitespace at the beginning of the string is skipped. If the receiver begins with a valid text representation of a floating-point number, that number's value is returned, otherwise 0.0 is returned. HUGE_VAL or -HUGE_VAL is returned on overflow. 0.0 is returned on underflow. Characters following the number are ignored.

- (float)**floatValue**
Returns the floating-point value of the receiver's text. Whitespace at the beginning of the string is skipped. If the receiver begins with a valid text representation of a floating-point number, that number's value is returned, otherwise 0.0 is returned. HUGE_VAL or -HUGE_VAL is returned on overflow. 0.0 is returned on underflow. Characters

following the number are ignored.

- (int)**intValue** Returns the integer value of the receiver's text. Whitespace at the beginning of the string is skipped. If the receiver begins with a valid representation of an integer, that number's value is returned, otherwise 0 is returned. INT_MAX or INT_MIN is returned on overflow. Characters following the number are ignored.

Working With Encodings

+ (NSStringEncoding)**defaultCStringEncoding** Returns the C string encoding assumed for any method accepting a C string as an argument.

- (BOOL)**canBeConvertedToEncoding:**(NSStringEncoding)*encoding*
Returns YES if the receiver can be converted to *encoding* without loss of information, and NO otherwise.

- (NSData *)**dataUsingEncoding:**(NSStringEncoding)*encoding*
Invokes **dataUsingEncoding:allowLossyConversion:** with NO as the argument to allow lossy conversion.

- (NSData *)**dataUsingEncoding:**(NSStringEncoding)*encoding*
allowLossyConversion:(BOOL)*flag* Returns an NSData object containing a representation of the receiver in *encoding*. If *flag* is NO and the receiver can't be converted without losing some information (such as accents or case) this method returns **nil**. If *flag* is YES and the receiver can't be converted without losing some information, some characters may be removed or altered in conversion.

- (NSStringEncoding)**fastestEncoding** Encoding in which this string can be expressed (with lossless conversion) most quickly.

- (NSStringEncoding)**smallestEncoding** Encoding in which this string can be expressed (with lossless conversion) in the most space efficient manner

Converting String Contents into a Property List

- (id)**propertyList** Depending on the format of the receiver's contents, returns a string, data, array, or dictionary object represention of those contents.

- (NSDictionary *)**propertyListFromStringsFileFormat**
Returns a dictionary object initialized with the keys and values found in the

receiver. The receiver's format must be that used for ª.stringº files.

Manipulating File System Paths

- (unsigned int)**completePathIntoString:**(NSString **)*outputName*
  caseSensitive:(BOOL)*flag*
  matchesIntoArray:(NSArray **)*outputArray*
  filterTypes:(NSArray *)*filterTypes*
  Regards the receiver as containing a partial filename and returns in *outputName* the longest matching path name. Case is considered if *flag* is YES. If *outputArray* is given, all matching filenames are return in *outputArray*. If *filterTypes* is provided, this method considers only those paths that match one of the types. Returns 0 if no matches are found; otherwise, the return value is positive.
- (NSString *)**lastPathComponent** Returns the last component of the receiver's path representation. Given the path ª/Foo/Bar.tiffº, this method returns a string containing ªBar.tiffº.
- (NSString *)**pathExtension** Returns the extension of the receiver's path representation. Given the path ª/Foo/Bar.tiffº, this method returns a string containing ªtiffº.
- (NSString *)**stringByAbbreviatingWithTildeInPath**
  Returns a string in which the user's home directory path is replace by ª~º.
- (NSString *)**stringByAppendingPathComponent:**(NSString *)*aString*
  Returns a string representing the receiver's path with the addition of the path component *aString*.
- (NSString *)**stringByAppendingPathExtension:**(NSString *)*aString*
  Returns a string representing the receiver's path with the addition of the extension *aString*.
- (NSString *)**stringByDeletingLastPathComponent**
  Returns the receiver's path representation minus the last component. Given the path ª/Foo/Bar.tiffº, this method returns a string containing ª/Fooº.
- (NSString *)**stringByDeletingPathExtension** Returns the receiver's path representation minus the extension on the last component. Given the path ª/Foo/Bar.tiffº, this method returns a string containing ª/Foo/Barº.
- (NSString *)**stringByExpandingTildeInPath** Returns a string in which a tilde is expanded to its full path equivalent.
- (NSString *)**stringByResolvingSymlinksInPath** Returns a string identical to the receiver's path except that any symbolic links have been resolved.
- (NSString *)**stringByStandardizingPath** Returns a string containing a ªstandardizedº path, one in which tildes are expanded and redundant elements (for example ª//º) eliminated.