SFML

* [Main Page](http://docs.google.com/index.htm)
* [Modules](http://docs.google.com/modules.htm)
* [Classes](http://docs.google.com/annotated.htm)
* [Files](http://docs.google.com/files.htm)
* [Class List](http://docs.google.com/annotated.htm)
* [Class Index](http://docs.google.com/classes.htm)
* [Class Hierarchy](http://docs.google.com/hierarchy.htm)
* [Class Members](http://docs.google.com/functions.htm)
* **sf**
* [String](http://docs.google.com/classsf_1_1String.htm)

[Public Types](#_gjdgxs) | [Public Member Functions](#_30j0zll) | [Static Public Attributes](#_1fob9te) | [Friends](#_3znysh7) | [Related Functions](#_2et92p0) | [List of all members](http://docs.google.com/classsf_1_1String-members.htm)

sf::String Class Reference

[System module](http://docs.google.com/group__system.htm)

Utility string class that automatically handles conversions between types and encodings. [More...](http://docs.google.com/classsf_1_1String.htm#details)

#include <[String.hpp](http://docs.google.com/String_8hpp_source.htm)>

| Public Types | |
| --- | --- |
| typedef std::basic\_string  < Uint32 >::iterator | [Iterator](http://docs.google.com/classsf_1_1String.htm#ac90f2b7b28f703020f8d027e98806235) |
|  | Iterator type. |
|  | |
| typedef std::basic\_string  < Uint32 >::const\_iterator | [ConstIterator](http://docs.google.com/classsf_1_1String.htm#a8e18efc2e8464f6eb82818902d527efa) |
|  | Constant iterator type. |
|  | |

| Public Member Functions | |
| --- | --- |
|  | [String](http://docs.google.com/classsf_1_1String.htm#a9563a4e93f692e0c8e8702b374ef8692) () |
|  | Default constructor. |
|  | |
|  | [String](http://docs.google.com/classsf_1_1String.htm#ac9df7f7696cff164794e338f3c89ccc5) (char ansiChar, const std::locale &locale=std::locale()) |
|  | Construct from a single ANSI character and a locale. |
|  | |
|  | [String](http://docs.google.com/classsf_1_1String.htm#aefaa202d2aa5ff85b4f75a5983367e86) (wchar\_t wideChar) |
|  | Construct from single wide character. |
|  | |
|  | [String](http://docs.google.com/classsf_1_1String.htm#a8e1a5027416d121187908e2ed77079ff) (Uint32 utf32Char) |
|  | Construct from single UTF-32 character. |
|  | |
|  | [String](http://docs.google.com/classsf_1_1String.htm#a57d2b8c289f9894f859564cad034bfc7) (const char \*ansiString, const std::locale &locale=std::locale()) |
|  | Construct from a null-terminated C-style ANSI string and a locale. |
|  | |
|  | [String](http://docs.google.com/classsf_1_1String.htm#a0aa41dcbd17b0c36c74d03d3b0147f1e) (const std::string &ansiString, const std::locale &locale=std::locale()) |
|  | Construct from an ANSI string and a locale. |
|  | |
|  | [String](http://docs.google.com/classsf_1_1String.htm#a5742d0a9b0c754f711820c2b5c40fa55) (const wchar\_t \*wideString) |
|  | Construct from null-terminated C-style wide string. |
|  | |
|  | [String](http://docs.google.com/classsf_1_1String.htm#a5e38151340af4f9a5f74ad24c0664074) (const std::wstring &wideString) |
|  | Construct from a wide string. |
|  | |
|  | [String](http://docs.google.com/classsf_1_1String.htm#aea3629adf19f9fe713d4946f6c75b214) (const Uint32 \*utf32String) |
|  | Construct from a null-terminated C-style UTF-32 string. |
|  | |
|  | [String](http://docs.google.com/classsf_1_1String.htm#a6eee86dbe75d16bbcc26e97416c2e1ca) (const std::basic\_string< Uint32 > &utf32String) |
|  | Construct from an UTF-32 string. |
|  | |
|  | [String](http://docs.google.com/classsf_1_1String.htm#af862594d3c4070d8ddbf08cf8dce4f59) (const [String](http://docs.google.com/classsf_1_1String.htm) &copy) |
|  | Copy constructor. |
|  | |
|  | [operator std::string](http://docs.google.com/classsf_1_1String.htm#a3664f92c3b488bf9f60f6b8eafe1844d) () const |
|  | Implicit cast operator to std::string (ANSI string) |
|  | |
|  | [operator std::wstring](http://docs.google.com/classsf_1_1String.htm#a20d50d45020d1dbe1779dda69ee83732) () const |
|  | Implicit cast operator to std::wstring (wide string) |
|  | |
| std::string | [toAnsiString](http://docs.google.com/classsf_1_1String.htm#a71d111ccfaf295b8b8be8ca4a3ceb5f4) (const std::locale &locale=std::locale()) const |
|  | Convert the unicode string to an ANSI string. |
|  | |
| std::wstring | [toWideString](http://docs.google.com/classsf_1_1String.htm#afd8b085ad7255543e4dc1285389d2d82) () const |
|  | Convert the unicode string to a wide string. |
|  | |
| [String](http://docs.google.com/classsf_1_1String.htm) & | [operator=](http://docs.google.com/classsf_1_1String.htm#a096255c066e5ef8c31952216b8ce9c42) (const [String](http://docs.google.com/classsf_1_1String.htm) &right) |
|  | Overload of assignment operator. |
|  | |
| [String](http://docs.google.com/classsf_1_1String.htm) & | [operator+=](http://docs.google.com/classsf_1_1String.htm#ae6563ce2c243ae2160eea8a354199f4e) (const [String](http://docs.google.com/classsf_1_1String.htm) &right) |
|  | Overload of += operator to append an UTF-32 string. |
|  | |
| Uint32 | [operator[]](http://docs.google.com/classsf_1_1String.htm#a649b89a65866bf03652f7b5cb6de3bb6) (std::size\_t index) const |
|  | Overload of [] operator to access a character by its position. |
|  | |
| Uint32 & | [operator[]](http://docs.google.com/classsf_1_1String.htm#abc989da7f4fb873ab29188d40772ab24) (std::size\_t index) |
|  | Overload of [] operator to access a character by its position. |
|  | |
| void | [clear](http://docs.google.com/classsf_1_1String.htm#a391c1b4950cbf3d3f8040cea73af2969) () |
|  | Clear the string. |
|  | |
| std::size\_t | [getSize](http://docs.google.com/classsf_1_1String.htm#a635d75c4cd830d5f639a41815dd0ce23) () const |
|  | Get the size of the string. |
|  | |
| bool | [isEmpty](http://docs.google.com/classsf_1_1String.htm#a6c43f0cbe84cf17fa6ba93d58b75fcdc) () const |
|  | Check whether the string is empty or not. |
|  | |
| void | [erase](http://docs.google.com/classsf_1_1String.htm#aaa78a0a46b3fbe200a4ccdedc326eb93) (std::size\_t position, std::size\_t count=1) |
|  | Erase one or more characters from the string. |
|  | |
| void | [insert](http://docs.google.com/classsf_1_1String.htm#ad0b1455deabf07af13ee79812e05fa02) (std::size\_t position, const [String](http://docs.google.com/classsf_1_1String.htm) &str) |
|  | Insert one or more characters into the string. |
|  | |
| std::size\_t | [find](http://docs.google.com/classsf_1_1String.htm#ae1fc8d6ced3b6082853f8013ea051b5c) (const [String](http://docs.google.com/classsf_1_1String.htm) &str, std::size\_t start=0) const |
|  | Find a sequence of one or more characters in the string. |
|  | |
| const Uint32 \* | [getData](http://docs.google.com/classsf_1_1String.htm#af2d4e70869ebd38e225c6796e1325ae4) () const |
|  | Get a pointer to the C-style array of characters. |
|  | |
| [Iterator](http://docs.google.com/classsf_1_1String.htm#ac90f2b7b28f703020f8d027e98806235) | [begin](http://docs.google.com/classsf_1_1String.htm#a8ec30ddc08e3a6bd11c99aed782f6dfe) () |
|  | Return an iterator to the beginning of the string. |
|  | |
| [ConstIterator](http://docs.google.com/classsf_1_1String.htm#a8e18efc2e8464f6eb82818902d527efa) | [begin](http://docs.google.com/classsf_1_1String.htm#a09bbf7704847ed35bf1c18aca6fba2a2) () const |
|  | Return an iterator to the beginning of the string. |
|  | |
| [Iterator](http://docs.google.com/classsf_1_1String.htm#ac90f2b7b28f703020f8d027e98806235) | [end](http://docs.google.com/classsf_1_1String.htm#ac823012f39cb6f61100418876e99d53b) () |
|  | Return an iterator to the beginning of the string. |
|  | |
| [ConstIterator](http://docs.google.com/classsf_1_1String.htm#a8e18efc2e8464f6eb82818902d527efa) | [end](http://docs.google.com/classsf_1_1String.htm#ab6fb6ee1b21bd5dbdc9d6840f035eb79) () const |
|  | Return an iterator to the beginning of the string. |
|  | |

| Static Public Attributes | |
| --- | --- |
| static const std::size\_t | [InvalidPos](http://docs.google.com/classsf_1_1String.htm#abaadecaf12a6b41c54d725c75fd28527) |
|  | Represents an invalid position in the string. |
|  | |

| Friends | |
| --- | --- |
| bool | **operator==** (const [String](http://docs.google.com/classsf_1_1String.htm) &left, const [String](http://docs.google.com/classsf_1_1String.htm) &right) |
|  | |
| bool | **operator<** (const [String](http://docs.google.com/classsf_1_1String.htm) &left, const [String](http://docs.google.com/classsf_1_1String.htm) &right) |
|  | |

| Related Functions | |
| --- | --- |
| (Note that these are not member functions.) | |
| bool | [operator==](http://docs.google.com/classsf_1_1String.htm#a483931724196c580552b68751fb4d837) (const [String](http://docs.google.com/classsf_1_1String.htm) &left, const [String](http://docs.google.com/classsf_1_1String.htm) &right) |
|  | Overload of == operator to compare two UTF-32 strings. |
|  | |
| bool | [operator!=](http://docs.google.com/classsf_1_1String.htm#a3bfb9217788a9978499b8d5696bb0ef2) (const [String](http://docs.google.com/classsf_1_1String.htm) &left, const [String](http://docs.google.com/classsf_1_1String.htm) &right) |
|  | Overload of != operator to compare two UTF-32 strings. |
|  | |
| bool | [operator<](http://docs.google.com/classsf_1_1String.htm#a5158a142e0966685ec7fb4e147b24ef0) (const [String](http://docs.google.com/classsf_1_1String.htm) &left, const [String](http://docs.google.com/classsf_1_1String.htm) &right) |
|  | Overload of < operator to compare two UTF-32 strings. |
|  | |
| bool | [operator>](http://docs.google.com/classsf_1_1String.htm#ac96278a8cbe282632b11f0c8c007df0c) (const [String](http://docs.google.com/classsf_1_1String.htm) &left, const [String](http://docs.google.com/classsf_1_1String.htm) &right) |
|  | Overload of > operator to compare two UTF-32 strings. |
|  | |
| bool | [operator<=](http://docs.google.com/classsf_1_1String.htm#ac1c1bb5dcf02aad3b2c0a1bf74a11cc9) (const [String](http://docs.google.com/classsf_1_1String.htm) &left, const [String](http://docs.google.com/classsf_1_1String.htm) &right) |
|  | Overload of <= operator to compare two UTF-32 strings. |
|  | |
| bool | [operator>=](http://docs.google.com/classsf_1_1String.htm#a112689eec28e0ca9489e8c4ec6a34493) (const [String](http://docs.google.com/classsf_1_1String.htm) &left, const [String](http://docs.google.com/classsf_1_1String.htm) &right) |
|  | Overload of >= operator to compare two UTF-32 strings. |
|  | |
| [String](http://docs.google.com/classsf_1_1String.htm) | [operator+](http://docs.google.com/classsf_1_1String.htm#af140f992b7698cf1448677c2c8e11bf1) (const [String](http://docs.google.com/classsf_1_1String.htm) &left, const [String](http://docs.google.com/classsf_1_1String.htm) &right) |
|  | Overload of binary + operator to concatenate two strings. |
|  | |

## Detailed Description

Utility string class that automatically handles conversions between types and encodings.

[sf::String](http://docs.google.com/classsf_1_1String.htm) is a utility string class defined mainly for convenience.

It is a Unicode string (implemented using UTF-32), thus it can store any character in the world (european, chinese, arabic, hebrew, etc.).

It automatically handles conversions from/to ANSI and wide strings, so that you can work with standard string classes and still be compatible with functions taking a [sf::String](http://docs.google.com/classsf_1_1String.htm).

[sf::String](http://docs.google.com/classsf_1_1String.htm) s;

std::string s1 = s; // automatically converted to ANSI string

std::wstring s2 = s; // automatically converted to wide string

s = "hello"; // automatically converted from ANSI string

s = L"hello"; // automatically converted from wide string

s += 'a'; // automatically converted from ANSI string

s += L'a'; // automatically converted from wide string

Conversions involving ANSI strings use the default user locale. However it is possible to use a custom locale if necessary:

std::locale locale;

[sf::String](http://docs.google.com/classsf_1_1String.htm) s;

...

std::string s1 = s.[toAnsiString](http://docs.google.com/classsf_1_1String.htm#a71d111ccfaf295b8b8be8ca4a3ceb5f4)(locale);

s = [sf::String](http://docs.google.com/classsf_1_1String.htm)("hello", locale);

[sf::String](http://docs.google.com/classsf_1_1String.htm) defines the most important functions of the standard std::string class: removing, random access, iterating, appending, comparing, etc. However it is a simple class provided for convenience, and you may have to consider using a more optimized class if your program requires complex string handling. The automatic conversion functions will then take care of converting your string to [sf::String](http://docs.google.com/classsf_1_1String.htm) whenever SFML requires it.

Please note that SFML also defines a low-level, generic interface for Unicode handling, see the [sf::Utf](http://docs.google.com/classsf_1_1Utf.htm) classes.

Definition at line [43](http://docs.google.com/String_8hpp_source.htm#l00043) of file [String.hpp](http://docs.google.com/String_8hpp_source.htm).

## Member Typedef Documentation

| typedef std::basic\_string<Uint32>::const\_iterator [sf::String::ConstIterator](http://docs.google.com/classsf_1_1String.htm#a8e18efc2e8464f6eb82818902d527efa) |
| --- |

Constant iterator type.

Definition at line [51](http://docs.google.com/String_8hpp_source.htm#l00051) of file [String.hpp](http://docs.google.com/String_8hpp_source.htm).

| typedef std::basic\_string<Uint32>::iterator [sf::String::Iterator](http://docs.google.com/classsf_1_1String.htm#ac90f2b7b28f703020f8d027e98806235) |
| --- |

Iterator type.

Definition at line [50](http://docs.google.com/String_8hpp_source.htm#l00050) of file [String.hpp](http://docs.google.com/String_8hpp_source.htm).

## Constructor & Destructor Documentation

| sf::String::String | ( |  | ) |  |
| --- | --- | --- | --- | --- |

Default constructor.

This constructor creates an empty string.

| sf::String::String | ( | char | *ansiChar*, |
| --- | --- | --- | --- |
|  |  | const std::locale & | *locale* = std::locale() |
|  | ) |  |  |

Construct from a single ANSI character and a locale.

The source character is converted to UTF-32 according to the given locale.

Parameters

| ansiChar | ANSI character to convert |
| --- | --- |
| locale | Locale to use for conversion |

| sf::String::String | ( | wchar\_t | *wideChar* | ) |  |
| --- | --- | --- | --- | --- | --- |

Construct from single wide character.

Parameters

| wideChar | Wide character to convert |
| --- | --- |

| sf::String::String | ( | Uint32 | *utf32Char* | ) |  |
| --- | --- | --- | --- | --- | --- |

Construct from single UTF-32 character.

Parameters

| utf32Char | UTF-32 character to convert |
| --- | --- |

| sf::String::String | ( | const char \* | *ansiString*, |
| --- | --- | --- | --- |
|  |  | const std::locale & | *locale* = std::locale() |
|  | ) |  |  |

Construct from a null-terminated C-style ANSI string and a locale.

The source string is converted to UTF-32 according to the given locale.

Parameters

| ansiString | ANSI string to convert |
| --- | --- |
| locale | Locale to use for conversion |

| sf::String::String | ( | const std::string & | *ansiString*, |
| --- | --- | --- | --- |
|  |  | const std::locale & | *locale* = std::locale() |
|  | ) |  |  |

Construct from an ANSI string and a locale.

The source string is converted to UTF-32 according to the given locale.

Parameters

| ansiString | ANSI string to convert |
| --- | --- |
| locale | Locale to use for conversion |

| sf::String::String | ( | const wchar\_t \* | *wideString* | ) |  |
| --- | --- | --- | --- | --- | --- |

Construct from null-terminated C-style wide string.

Parameters

| wideString | Wide string to convert |
| --- | --- |

| sf::String::String | ( | const std::wstring & | *wideString* | ) |  |
| --- | --- | --- | --- | --- | --- |

Construct from a wide string.

Parameters

| wideString | Wide string to convert |
| --- | --- |

| sf::String::String | ( | const Uint32 \* | *utf32String* | ) |  |
| --- | --- | --- | --- | --- | --- |

Construct from a null-terminated C-style UTF-32 string.

Parameters

| utf32String | UTF-32 string to assign |
| --- | --- |

| sf::String::String | ( | const std::basic\_string< Uint32 > & | *utf32String* | ) |  |
| --- | --- | --- | --- | --- | --- |

Construct from an UTF-32 string.

Parameters

| utf32String | UTF-32 string to assign |
| --- | --- |

| sf::String::String | ( | const [String](http://docs.google.com/classsf_1_1String.htm) & | *copy* | ) |  |
| --- | --- | --- | --- | --- | --- |

Copy constructor.

Parameters

| copy | Instance to copy |
| --- | --- |

## Member Function Documentation

| [Iterator](http://docs.google.com/classsf_1_1String.htm#ac90f2b7b28f703020f8d027e98806235) sf::String::begin | ( |  | ) |  |
| --- | --- | --- | --- | --- |

Return an iterator to the beginning of the string.

ReturnsRead-write iterator to the beginning of the string characters See Also[end](http://docs.google.com/classsf_1_1String.htm#ac823012f39cb6f61100418876e99d53b)

| [ConstIterator](http://docs.google.com/classsf_1_1String.htm#a8e18efc2e8464f6eb82818902d527efa) sf::String::begin | ( |  | ) | const |
| --- | --- | --- | --- | --- |

Return an iterator to the beginning of the string.

ReturnsRead-only iterator to the beginning of the string characters See Also[end](http://docs.google.com/classsf_1_1String.htm#ac823012f39cb6f61100418876e99d53b)

| void sf::String::clear | ( |  | ) |  |
| --- | --- | --- | --- | --- |

Clear the string.

This function removes all the characters from the string.

See Also[isEmpty](http://docs.google.com/classsf_1_1String.htm#a6c43f0cbe84cf17fa6ba93d58b75fcdc), [erase](http://docs.google.com/classsf_1_1String.htm#aaa78a0a46b3fbe200a4ccdedc326eb93)

| [Iterator](http://docs.google.com/classsf_1_1String.htm#ac90f2b7b28f703020f8d027e98806235) sf::String::end | ( |  | ) |  |
| --- | --- | --- | --- | --- |

Return an iterator to the beginning of the string.

The end iterator refers to 1 position past the last character; thus it represents an invalid character and should never be accessed.

ReturnsRead-write iterator to the end of the string characters See Also[begin](http://docs.google.com/classsf_1_1String.htm#a8ec30ddc08e3a6bd11c99aed782f6dfe)

| [ConstIterator](http://docs.google.com/classsf_1_1String.htm#a8e18efc2e8464f6eb82818902d527efa) sf::String::end | ( |  | ) | const |
| --- | --- | --- | --- | --- |

Return an iterator to the beginning of the string.

The end iterator refers to 1 position past the last character; thus it represents an invalid character and should never be accessed.

ReturnsRead-only iterator to the end of the string characters See Also[begin](http://docs.google.com/classsf_1_1String.htm#a8ec30ddc08e3a6bd11c99aed782f6dfe)

| void sf::String::erase | ( | std::size\_t | *position*, |
| --- | --- | --- | --- |
|  |  | std::size\_t | *count* = 1 |
|  | ) |  |  |

Erase one or more characters from the string.

This function removes a sequence of *count* characters starting from *position*.

Parameters

| position | Position of the first character to erase |
| --- | --- |
| count | Number of characters to erase |

| std::size\_t sf::String::find | ( | const [String](http://docs.google.com/classsf_1_1String.htm) & | *str*, |
| --- | --- | --- | --- |
|  |  | std::size\_t | *start* = 0 |
|  | ) |  | const |

Find a sequence of one or more characters in the string.

This function searches for the characters of *str* into the string, starting from *start*.

Parameters

| str | Characters to find |
| --- | --- |
| start | Where to begin searching |

ReturnsPosition of *str* in the string, or [String::InvalidPos](http://docs.google.com/classsf_1_1String.htm#abaadecaf12a6b41c54d725c75fd28527) if not found

| const Uint32\* sf::String::getData | ( |  | ) | const |
| --- | --- | --- | --- | --- |

Get a pointer to the C-style array of characters.

This functions provides a read-only access to a null-terminated C-style representation of the string. The returned pointer is temporary and is meant only for immediate use, thus it is not recommended to store it.

ReturnsRead-only pointer to the array of characters

| std::size\_t sf::String::getSize | ( |  | ) | const |
| --- | --- | --- | --- | --- |

Get the size of the string.

ReturnsNumber of characters in the string See Also[isEmpty](http://docs.google.com/classsf_1_1String.htm#a6c43f0cbe84cf17fa6ba93d58b75fcdc)

| void sf::String::insert | ( | std::size\_t | *position*, |
| --- | --- | --- | --- |
|  |  | const [String](http://docs.google.com/classsf_1_1String.htm) & | *str* |
|  | ) |  |  |

Insert one or more characters into the string.

This function inserts the characters of *str* into the string, starting from *position*.

Parameters

| position | Position of insertion |
| --- | --- |
| str | Characters to insert |

| bool sf::String::isEmpty | ( |  | ) | const |
| --- | --- | --- | --- | --- |

Check whether the string is empty or not.

ReturnsTrue if the string is empty (i.e. contains no character) See Also[clear](http://docs.google.com/classsf_1_1String.htm#a391c1b4950cbf3d3f8040cea73af2969), [getSize](http://docs.google.com/classsf_1_1String.htm#a635d75c4cd830d5f639a41815dd0ce23)

| sf::String::operator std::string | ( |  | ) | const |
| --- | --- | --- | --- | --- |

Implicit cast operator to std::string (ANSI string)

The current global locale is used for conversion. If you want to explicitely specify a locale, see toAnsiString. Characters that do not fit in the target encoding are discarded from the returned string. This operator is defined for convenience, and is equivalent to calling [toAnsiString()](http://docs.google.com/classsf_1_1String.htm#a71d111ccfaf295b8b8be8ca4a3ceb5f4).

ReturnsConverted ANSI string See Also[toAnsiString](http://docs.google.com/classsf_1_1String.htm#a71d111ccfaf295b8b8be8ca4a3ceb5f4), operator std::wstring

| sf::String::operator std::wstring | ( |  | ) | const |
| --- | --- | --- | --- | --- |

Implicit cast operator to std::wstring (wide string)

Characters that do not fit in the target encoding are discarded from the returned string. This operator is defined for convenience, and is equivalent to calling [toWideString()](http://docs.google.com/classsf_1_1String.htm#afd8b085ad7255543e4dc1285389d2d82).

ReturnsConverted wide string See Also[toWideString](http://docs.google.com/classsf_1_1String.htm#afd8b085ad7255543e4dc1285389d2d82), operator std::string

| [String](http://docs.google.com/classsf_1_1String.htm)& sf::String::operator+= | ( | const [String](http://docs.google.com/classsf_1_1String.htm) & | *right* | ) |  |
| --- | --- | --- | --- | --- | --- |

Overload of += operator to append an UTF-32 string.

Parameters

| right | [String](http://docs.google.com/classsf_1_1String.htm) to append |
| --- | --- |

ReturnsReference to self

| [String](http://docs.google.com/classsf_1_1String.htm)& sf::String::operator= | ( | const [String](http://docs.google.com/classsf_1_1String.htm) & | *right* | ) |  |
| --- | --- | --- | --- | --- | --- |

Overload of assignment operator.

Parameters

| right | Instance to assign |
| --- | --- |

ReturnsReference to self

| Uint32 sf::String::operator[] | ( | std::size\_t | *index* | ) | const |
| --- | --- | --- | --- | --- | --- |

Overload of [] operator to access a character by its position.

This function provides read-only access to characters. Note: this function doesn't throw if *index* is out of range.

Parameters

| index | Index of the character to get |
| --- | --- |

ReturnsCharacter at position *index*

| Uint32& sf::String::operator[] | ( | std::size\_t | *index* | ) |  |
| --- | --- | --- | --- | --- | --- |

Overload of [] operator to access a character by its position.

This function provides read and write access to characters. Note: this function doesn't throw if *index* is out of range.

Parameters

| index | Index of the character to get |
| --- | --- |

ReturnsReference to the character at position *index*

| std::string sf::String::toAnsiString | ( | const std::locale & | *locale* = std::locale() | ) | const |
| --- | --- | --- | --- | --- | --- |

Convert the unicode string to an ANSI string.

The UTF-32 string is converted to an ANSI string in the encoding defined by *locale*. Characters that do not fit in the target encoding are discarded from the returned string.

Parameters

| locale | Locale to use for conversion |
| --- | --- |

ReturnsConverted ANSI string See Also[toWideString](http://docs.google.com/classsf_1_1String.htm#afd8b085ad7255543e4dc1285389d2d82), operator std::string

| std::wstring sf::String::toWideString | ( |  | ) | const |
| --- | --- | --- | --- | --- |

Convert the unicode string to a wide string.

Characters that do not fit in the target encoding are discarded from the returned string.

ReturnsConverted wide string See Also[toAnsiString](http://docs.google.com/classsf_1_1String.htm#a71d111ccfaf295b8b8be8ca4a3ceb5f4), operator std::wstring

## Friends And Related Function Documentation

| | bool operator!= | ( | const [String](http://docs.google.com/classsf_1_1String.htm) & | *left*, | | --- | --- | --- | --- | |  |  | const [String](http://docs.google.com/classsf_1_1String.htm) & | *right* | |  | ) |  |  | | related |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |

Overload of != operator to compare two UTF-32 strings.

Parameters

| left | Left operand (a string) |
| --- | --- |
| right | Right operand (a string) |

ReturnsTrue if both strings are different

| | [String](http://docs.google.com/classsf_1_1String.htm) operator+ | ( | const [String](http://docs.google.com/classsf_1_1String.htm) & | *left*, | | --- | --- | --- | --- | |  |  | const [String](http://docs.google.com/classsf_1_1String.htm) & | *right* | |  | ) |  |  | | related |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |

Overload of binary + operator to concatenate two strings.

Parameters

| left | Left operand (a string) |
| --- | --- |
| right | Right operand (a string) |

ReturnsConcatenated string

| | bool operator< | ( | const [String](http://docs.google.com/classsf_1_1String.htm) & | *left*, | | --- | --- | --- | --- | |  |  | const [String](http://docs.google.com/classsf_1_1String.htm) & | *right* | |  | ) |  |  | | related |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |

Overload of < operator to compare two UTF-32 strings.

Parameters

| left | Left operand (a string) |
| --- | --- |
| right | Right operand (a string) |

ReturnsTrue if *left* is alphabetically lesser than *right*

| | bool operator<= | ( | const [String](http://docs.google.com/classsf_1_1String.htm) & | *left*, | | --- | --- | --- | --- | |  |  | const [String](http://docs.google.com/classsf_1_1String.htm) & | *right* | |  | ) |  |  | | related |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |

Overload of <= operator to compare two UTF-32 strings.

Parameters

| left | Left operand (a string) |
| --- | --- |
| right | Right operand (a string) |

ReturnsTrue if *left* is alphabetically lesser or equal than *right*

| | bool operator== | ( | const [String](http://docs.google.com/classsf_1_1String.htm) & | *left*, | | --- | --- | --- | --- | |  |  | const [String](http://docs.google.com/classsf_1_1String.htm) & | *right* | |  | ) |  |  | | related |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |

Overload of == operator to compare two UTF-32 strings.

Parameters

| left | Left operand (a string) |
| --- | --- |
| right | Right operand (a string) |

ReturnsTrue if both strings are equal

| | bool operator> | ( | const [String](http://docs.google.com/classsf_1_1String.htm) & | *left*, | | --- | --- | --- | --- | |  |  | const [String](http://docs.google.com/classsf_1_1String.htm) & | *right* | |  | ) |  |  | | related |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |

Overload of > operator to compare two UTF-32 strings.

Parameters

| left | Left operand (a string) |
| --- | --- |
| right | Right operand (a string) |

ReturnsTrue if *left* is alphabetically greater than *right*

| | bool operator>= | ( | const [String](http://docs.google.com/classsf_1_1String.htm) & | *left*, | | --- | --- | --- | --- | |  |  | const [String](http://docs.google.com/classsf_1_1String.htm) & | *right* | |  | ) |  |  | | related |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |

Overload of >= operator to compare two UTF-32 strings.

Parameters

| left | Left operand (a string) |
| --- | --- |
| right | Right operand (a string) |

ReturnsTrue if *left* is alphabetically greater or equal than *right*

## Member Data Documentation

| | const std::size\_t sf::String::InvalidPos | | --- | | static |
| --- | --- | --- |

Represents an invalid position in the string.

Definition at line [56](http://docs.google.com/String_8hpp_source.htm#l00056) of file [String.hpp](http://docs.google.com/String_8hpp_source.htm).

The documentation for this class was generated from the following file:

* [String.hpp](http://docs.google.com/String_8hpp_source.htm)

Copyright � Laurent Gomila  ::  Documentation generated by [doxygen](http://www.doxygen.org/)  ::