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)
* [File List](http://docs.google.com/files.htm)
* [include](http://docs.google.com/dir_f3190241575fd2bd132a392ae6942f4a.htm)
* [SFML](http://docs.google.com/dir_692f376662c82a26cfe4cfa3aceebe24.htm)
* [System](http://docs.google.com/dir_60c5c649f8df3b69a45a020d59f81335.htm)

String.hpp

1

2 //

3 // SFML - Simple and Fast Multimedia Library

4 // Copyright (C) 2007-2013 Laurent Gomila (laurent.gom@gmail.com)

5 //

6 // This software is provided 'as-is', without any express or implied warranty.

7 // In no event will the authors be held liable for any damages arising from the use of this software.

8 //

9 // Permission is granted to anyone to use this software for any purpose,

10 // including commercial applications, and to alter it and redistribute it freely,

11 // subject to the following restrictions:

12 //

13 // 1. The origin of this software must not be misrepresented;

14 // you must not claim that you wrote the original software.

15 // If you use this software in a product, an acknowledgment

16 // in the product documentation would be appreciated but is not required.

17 //

18 // 2. Altered source versions must be plainly marked as such,

19 // and must not be misrepresented as being the original software.

20 //

21 // 3. This notice may not be removed or altered from any source distribution.

22 //

24

25 #ifndef SFML\_STRING\_HPP

26 #define SFML\_STRING\_HPP

27

29 // Headers

31 #include <SFML/System/Export.hpp>

32 #include <locale>

33 #include <string>

34

35

36 namespace sf

37 {

[43](http://docs.google.com/classsf_1_1String.htm) class SFML\_SYSTEM\_API [String](http://docs.google.com/classsf_1_1String.htm)

44 {

45 public :

46

48  // Types

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

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

52

54  // Static member data

[56](http://docs.google.com/classsf_1_1String.htm#abaadecaf12a6b41c54d725c75fd28527)  static const std::size\_t [InvalidPos](http://docs.google.com/classsf_1_1String.htm#abaadecaf12a6b41c54d725c75fd28527);

57

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

65

76  [String](http://docs.google.com/classsf_1_1String.htm)(char ansiChar, const std::locale& locale = std::locale());

77

84  [String](http://docs.google.com/classsf_1_1String.htm)(wchar\_t wideChar);

85

92  [String](http://docs.google.com/classsf_1_1String.htm)(Uint32 utf32Char);

93

104  [String](http://docs.google.com/classsf_1_1String.htm)(const char\* ansiString, const std::locale& locale = std::locale());

105

116  [String](http://docs.google.com/classsf_1_1String.htm)(const std::string& ansiString, const std::locale& locale = std::locale());

117

124  [String](http://docs.google.com/classsf_1_1String.htm)(const wchar\_t\* wideString);

125

132  [String](http://docs.google.com/classsf_1_1String.htm)(const std::wstring& wideString);

133

140  [String](http://docs.google.com/classsf_1_1String.htm)(const Uint32\* utf32String);

141

148  [String](http://docs.google.com/classsf_1_1String.htm)(const std::basic\_string<Uint32>& utf32String);

149

156  [String](http://docs.google.com/classsf_1_1String.htm)(const [String](http://docs.google.com/classsf_1_1String.htm)& copy);

157

173  operator std::string() const;

174

188  operator std::wstring() const;

189

205  std::string toAnsiString(const std::locale& locale = std::locale()) const;

206

218  std::wstring toWideString() const;

219

228  [String](http://docs.google.com/classsf_1_1String.htm)& operator =(const [String](http://docs.google.com/classsf_1_1String.htm)& right);

229

238  [String](http://docs.google.com/classsf_1_1String.htm)& operator +=(const [String](http://docs.google.com/classsf_1_1String.htm)& right);

239

251  Uint32 operator [](std::size\_t index) const;

252

264  Uint32& operator [](std::size\_t index);

265

274  void clear();

275

284  std::size\_t getSize() const;

285

294  bool isEmpty() const;

295

306  void erase(std::size\_t position, std::size\_t count = 1);

307

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

319

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

333

345  const Uint32\* getData() const;

346

355  [Iterator](http://docs.google.com/classsf_1_1String.htm#ac90f2b7b28f703020f8d027e98806235) begin();

356

365  [ConstIterator](http://docs.google.com/classsf_1_1String.htm#a8e18efc2e8464f6eb82818902d527efa) begin() const;

366

379  [Iterator](http://docs.google.com/classsf_1_1String.htm#ac90f2b7b28f703020f8d027e98806235) end();

380

393  [ConstIterator](http://docs.google.com/classsf_1_1String.htm#a8e18efc2e8464f6eb82818902d527efa) end() const;

394

395 private :

396

397  friend SFML\_SYSTEM\_API bool operator ==(const [String](http://docs.google.com/classsf_1_1String.htm)& left, const [String](http://docs.google.com/classsf_1_1String.htm)& right);

398  friend SFML\_SYSTEM\_API bool operator <(const [String](http://docs.google.com/classsf_1_1String.htm)& left, const [String](http://docs.google.com/classsf_1_1String.htm)& right);

399

401  // Member data

403  std::basic\_string<Uint32> m\_string;

404 };

405

416 SFML\_SYSTEM\_API bool operator ==(const [String](http://docs.google.com/classsf_1_1String.htm)& left, const [String](http://docs.google.com/classsf_1_1String.htm)& right);

417

428 SFML\_SYSTEM\_API bool operator !=(const [String](http://docs.google.com/classsf_1_1String.htm)& left, const [String](http://docs.google.com/classsf_1_1String.htm)& right);

429

440 SFML\_SYSTEM\_API bool operator <(const [String](http://docs.google.com/classsf_1_1String.htm)& left, const [String](http://docs.google.com/classsf_1_1String.htm)& right);

441

452 SFML\_SYSTEM\_API bool operator >(const [String](http://docs.google.com/classsf_1_1String.htm)& left, const [String](http://docs.google.com/classsf_1_1String.htm)& right);

453

464 SFML\_SYSTEM\_API bool operator <=(const [String](http://docs.google.com/classsf_1_1String.htm)& left, const [String](http://docs.google.com/classsf_1_1String.htm)& right);

465

476 SFML\_SYSTEM\_API bool operator >=(const [String](http://docs.google.com/classsf_1_1String.htm)& left, const [String](http://docs.google.com/classsf_1_1String.htm)& right);

477

488 SFML\_SYSTEM\_API [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);

489

490 } // namespace sf

491

492

493 #endif // SFML\_STRING\_HPP

494

495

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