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**
* [Lock](http://docs.google.com/classsf_1_1Lock.htm)

[Public Member Functions](#_gjdgxs) | [List of all members](http://docs.google.com/classsf_1_1Lock-members.htm)

sf::Lock Class Reference

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

Automatic wrapper for locking and unlocking mutexes. [More...](http://docs.google.com/classsf_1_1Lock.htm#details)

#include <[Lock.hpp](http://docs.google.com/Lock_8hpp_source.htm)>

Inheritance diagram for sf::Lock:



| Public Member Functions | |
| --- | --- |
|  | [Lock](http://docs.google.com/classsf_1_1Lock.htm#a1a4c5d7a15da61103d85c9aa7f118920) ([Mutex](http://docs.google.com/classsf_1_1Mutex.htm) &mutex) |
|  | Construct the lock with a target mutex. |
|  | |
|  | [~Lock](http://docs.google.com/classsf_1_1Lock.htm#a8168b36323a18ccf5b6bc531d964aec5) () |
|  | Destructor. |
|  | |

## Detailed Description

Automatic wrapper for locking and unlocking mutexes.

[sf::Lock](http://docs.google.com/classsf_1_1Lock.htm) is a RAII wrapper for [sf::Mutex](http://docs.google.com/classsf_1_1Mutex.htm).

By unlocking it in its destructor, it ensures that the mutex will always be released when the current scope (most likely a function) ends. This is even more important when an exception or an early return statement can interrupt the execution flow of the function.

For maximum robustness, [sf::Lock](http://docs.google.com/classsf_1_1Lock.htm) should always be used to lock/unlock a mutex.

Usage example:

[sf::Mutex](http://docs.google.com/classsf_1_1Mutex.htm) mutex;

void function()

{

[sf::Lock](http://docs.google.com/classsf_1_1Lock.htm) lock(mutex); // mutex is now locked

functionThatMayThrowAnException(); // mutex is unlocked if this function throws

if (someCondition)

return; // mutex is unlocked

} // mutex is unlocked

Because the mutex is not explicitely unlocked in the code, it may remain locked longer than needed. If the region of the code that needs to be protected by the mutex is not the entire function, a good practice is to create a smaller, inner scope so that the lock is limited to this part of the code.

[sf::Mutex](http://docs.google.com/classsf_1_1Mutex.htm) mutex;

void function()

{

{

[sf::Lock](http://docs.google.com/classsf_1_1Lock.htm) lock(mutex);

codeThatRequiresProtection();

} // mutex is unlocked here

codeThatDoesntCareAboutTheMutex();

}

Having a mutex locked longer than required is a bad practice which can lead to bad performances. Don't forget that when a mutex is locked, other threads may be waiting doing nothing until it is released.

See Also[sf::Mutex](http://docs.google.com/classsf_1_1Mutex.htm)

Definition at line [43](http://docs.google.com/Lock_8hpp_source.htm#l00043) of file [Lock.hpp](http://docs.google.com/Lock_8hpp_source.htm).

## Constructor & Destructor Documentation

| | sf::Lock::Lock | ( | [Mutex](http://docs.google.com/classsf_1_1Mutex.htm) & | *mutex* | ) |  | | --- | --- | --- | --- | --- | --- | | explicit |
| --- | --- | --- | --- | --- | --- | --- | --- |

Construct the lock with a target mutex.

The mutex passed to [sf::Lock](http://docs.google.com/classsf_1_1Lock.htm) is automatically locked.

Parameters

| mutex | [Mutex](http://docs.google.com/classsf_1_1Mutex.htm) to lock |
| --- | --- |

| sf::Lock::~Lock | ( |  | ) |  |
| --- | --- | --- | --- | --- |

Destructor.

The destructor of [sf::Lock](http://docs.google.com/classsf_1_1Lock.htm) automatically unlocks its mutex.

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

* [Lock.hpp](http://docs.google.com/Lock_8hpp_source.htm)

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