The Null-Pointer-Literal Keyword

The **nullptr** literal, unlike `0` or `NULL`, ~~unambiguously denotes a *null-address* value~~.

Description

The **nullptr** keyword is a *prvalue* (pure *rvalue*) of type `std::nullptr_t` ~~representing the implementation-defined bit pattern corresponding to a **null address** on the host platform~~; **nullptr** and other values of type `std::nullptr_t`, along with the integer literal `0` and the macro `NULL`, can be converted implicitly to any pointer or pointer-to-member type:

```
#include <cstddef> // NULL
int data;  // nonmember data

int* pi0 = &data;    // Initialize with non-null address.
int* pi1 = nullptr;  // Initialize with null address.
int* pi2 = NULL;     //      "       "    "     "
int* pi3 = 0;        //      "       "    "     "

double f(int x);  // nonmember function

double (*pf0)(int) = &f;       // Initialize with non-null address.
double (*pf1)(int) = nullptr;  // Initialize with null address.

struct S
{
    short d_data;    // member data
    float g(int y);  // member function
};

short S::*pmd0 = &S::d_data;  // Initialize with non-null address.
short S::*pmd1 = nullptr;     // Initialize with null address.

float (S::*pmf0)(int) = &S::g;    // Initialize with non-null address.
float (S::*pmf1)(int) = nullptr;  // Initialize with null address.
```