## Attributes in C++11/C++14

Let's take a look at specific attributes in C++ 11 and attribute additions in C++ 14.

### WE'LL COVER THE FOLLOWING

- Specifying Annotations in C++11
  - Attributes in C++11
  - C++14 Additions
    - Example of [[deprecated]]:

# Specifying Annotations in C++11 #

C++11 took one big step to minimize the need to use vendor specific syntax. By introducing the standard format, we can move a lot of compiler-specific attributes to the universal set.

C++11 provides a cleaner format of specifying annotations over our code.

```
The basic syntax is just [[attr]] or [[namespace::attr]].
```

You can use [[attr]] over almost anything: types, functions, enums, etc., etc.

For example:

## Attributes in C++11 #

## [[noreturn]]:

It tells the compiler that control flow will not return to the caller. Examples:

- [[noreturn]] void terminate() noexcept;
- functions like std::abort or std::exit are also marked with this

attribute.

### [[carries\_dependency]]:

Indicates that the dependency chain in release-consume std::memory\_order
propagates in and out of the function, which allows the compiler to skip
unnecessary memory fence instructions. Mostly to help to optimise multithreaded code and when using different memory models.

### C++14 Additions #

## [[deprecated]] and [[deprecated("reason")]]:

Code marked with this attribute will be reported by the compiler. You can set a "**reason**" why.

Example of [[deprecated]]: #

```
#include <iostream>
using namespace std;

[[deprecated("use AwesomeFunc instead")]] void GoodFunc() { }

int main() {
    GoodFunc();
}
```

GCC might report the following warning when you run above the code:

```
warning: 'void GoodFunc()' is deprecated: use AwesomeFunc instead
[-Wdeprecated-declarations]
```

Now that you know a bit about the old approach, we will learn about how C++17 deals with it in the next lesson.

This was all for the attributes in C++ 11 and 14. The next lesson will now introduce the new attributes present in C++ 17.