How to Ignore [[nodiscard]]

In some circumstances, you may not want the [[nodiscard]] functionality. There are methods of ignoring its warnings. We'll discuss them below.

WE'LL COVER THE FOLLOWING

With `[[nodiscard]]` you should use the returned value - by assigning it to a variable or by using it directly. If you forget, you'll get an "unused variable" warning.

With an Attribute

There are situations where you might want to suppress such a warning. To do that you can use another attribute from C++17: `[[maybe_unused]]`:

```
[[nodiscard]] int Compute() { return 42; }
[[maybe_unused]] auto t = Compute();
```

Casting to Void

Also, as mentioned in the Attributes Chapter, you can cast the function call to `void`, and the compiler will think you "used" the value:

```
[[nodiscard]] int Compute();
static_cast<void>(Compute()); // used
```

Separate Function

Separate Function

Another good alternative might be to write a separate function that wraps the results and pretends to use it:

```
template <class T> inline void discard_on_purpose(T&&) {}

// use case:
discard_on_purpose(Compute());
```

Playground

Have a go at executing all the above methods.

```
[[nodiscard]] int Compute() { return 42; }

template <class T> inline void discard_on_purpose(T&&) { }

int main()
{
    // with an attribute:
    [[maybe_unused]] auto t = Compute();

    // separate function:
    discard_on_purpose(Compute());

    // casting to void:
    static_cast<void>(Compute());
    return 0;
}
```

> **Note**: Be careful with the techniques to avoid warnings with `[[nodiscard]]`. It’s better to follow the rules of the attribute rather than artificially prevent them.

Before C++17

Most of the attributes that went into the standardised `[[attrib]]` come from compiler extensions, same happened with `[[nodiscard]]`.

For example, in GCC/Clang, there’s: `__attribute__((warn_unused_result))`

MSVC offers `_Check_return_` - see at MSDN: Annotating Function Behavior.

Our talk on `[[nodiscard]]` has come to an end. A general summary of this attribute can be found in the next lesson.