C++17 additions

Fallthrough, maybe_unused and nodiscard are the three new attributes in C++ 17 that this lesson talks about.

WE'LL COVER THE FOLLOWING

- Attributes in C++17
  - Two new attributes
    - [[fallthrough]] attribute
    - [[maybe_unused]] attribute

Attributes in C++17

With C++17 we get three more standard attributes:

- `[[fallthrough]]`
- `[[maybe_unused]]`
- `[[nodiscard]]`

> *Extra Info:* The new attributes were specified in P0188 and P0068(reasoning).

Plus *three* supporting features:

- Attributes for Namespaces and Enumerators
- Ignore Unknown Attributes
- Using Attribute Namespaces Without Repetition

Let's go through the new attributes first.

Two new attributes

[[fallthrough]] attribute

Indicates that a fall-through in a switch statement is intentional and a warning should not be issued for it.

Example:

```
switch (c)
{
  case 'a':
    f(); // Warning! fallthrough is perhaps a programmer error
  case 'b':
    g();
    [[fallthrough]]; // Warning suppressed, fallthrough is ok
  case 'c':
    h();
}
```

With this attribute, the compiler can understand the intentions of a programmer. And what's more, it's also much more readable than using a comment.

[[maybe_unused]] attribute

Suppresses compiler warnings about unused entities:

Example:

```
static void impl1() { ... } // Compilers may warn about this

[[maybe_unused]] static void impl2() { ... } // Warning suppressed

void foo()
{
  int x = 42; // Compilers may warn when x is not used later
  [[maybe_unused]] int y = 42; // Warning suppressed for y
}
```

Such behaviour is helpful when some of the variables and functions are used in debug only path. For example in `assert()` macros;

```
void doSomething(std::string_view a, std::string_view b) {
    assert(a.size() < b.size());
}
```

If later `a` or `b` is no used in this function, then the compiler will generate a warning in release only builds. Marking the given argument with `[[maybe_unused]]` will solve this warning.

We have discussed two of the three standard attributes introduced in C++ 17. The next lesson will now elaborate on the third attribute.