static_assert With No Message

C++17 has built upon the previous functionality of 'static_assert' to provide a more convenient experience.

WE'LL COVER THE FOLLOWING

static_assert

This feature adds a new overload for `static_assert`. It enables you to have the condition inside `static_assert` without passing the message.

It will be compatible with other asserts like `BOOST_STATIC_ASSERT`. Programmers with boost experience will now have no trouble switching to C++17 static_assert.

```
static_assert(std::is_arithmetic_v<T>, "T must be arithmetic");
static_assert(std::is_arithmetic_v<T>); // no message needed since C++17
```

In many cases, the condition you check is expressive enough and doesn't need to be mentioned in the message string.

> *Extra Info:* The change was proposed in: N3928.

Fix: Different `begin` and `end` Types In Range-Based For Loop

C++11 added range-based for loops:

```
for (for-range-declaration : for-range-initializer)
{

  statement;
}
```

According to the C++14 standard such loop expression was equivalent to the following code:

```
{
  auto && __range = for-range-initializer;
  for ( auto __begin = begin-expr, __end = end-expr; __begin != __end; ++__begin )
  {
    for-range-declaration = *__begin;
    statement
  }
}
```

As you can see, `__begin` and `__end` have the same type. This works nicely but is not scalable enough.

For example, you might want to iterate until some sentinel value with a different type than the start of the range.

In C++17 range-based for loops are defined as equivalent to the following code:

```
{
  auto && __range = for-range-initializer;
  auto __begin = begin-expr;
  auto __end = end-expr;
  for ( ; __begin != __end; ++__begin ) {
    for-range-declaration = *__begin;
    statement
  }
}
```

Types of `__begin` and `__end` might be different; only the comparison operator is required. That change has no effect on existing for loops but it provides more options for libraries. For example, this little change allows Range TS (and Ranges in C++20) to work with the range-based for loop.

> *Extra Info:* The change was proposed in: P0184R0.

We will conclude with compiler support, in the next lesson.