Call Operator

Let's take a deeper dive into what function objects are.

WE'LL COVER THE FOLLOWING

By overloading the function call operator, `()`, we can call objects like ordinary function objects that may or may not have arguments. These special objects are known as **function objects** or, wrongly, as **functors**.

The best feature of function objects is that they can have a *state*. Since they are objects, data is stored inside them, but they can also be used as functions to return data.

Functors

Functors are very similar to lambda functions. We can say that lambdas actually create anonymous functors.

Because of this, functors are used frequently in STL algorithms as arguments. These functors can then be applied to the data being passed to the STL functions.

`std::add`, `std::transform`, and `std::reduce` are just a few of the functions that can use functors and apply them to data.

- A functor that takes a single argument is a **unary functor**.
- A functor that takes two arguments is a **binary functor**.

Further information

- lambda functions

We will look at the example of the Call operator in the next lesson.