A Basic Example Of std::optional

Let's look at std::optional in action.

Here’s a simple example of what you can do with optional:

```
// UI Class...
std::optional<std::string> UI::FindUserNick()
{
  if (IsNickAvailable())
    return mStrNickName; // return a string

  return std::nullopt; // same as return { };
}

// use:
std::optional<std::string> UserNick = UI->FindUserNick();
if (UserNick)
  Show(*UserNick);
```

In the above code, we define a function that returns an `optional` containing a `string`. If the user’s nickname is available, then it will return a string. If not, then it returns `nullopt`. Later we can assign it to an optional and check (it converts to `bool`) if it contains any value or not. Optional defines `operator*` so we can easily access the stored value.

In the following sections, you’ll see how to create `std::optional`, operate on it, pass it around, and even what is the performance cost you might want to consider.