## **Examples of std::variant**

This lesson explains the basic idea of ErrorCode.

```
we'll cover the following ^
• Error Handling
```

Having learned most of the std::variant details, we can now explore a few examples.

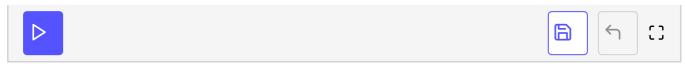
## **Error Handling** #

The basic idea is to wrap the possible return type with some ErrorCode, and that way allow functions to output more information about the errors.

Without using exceptions or output parameters.

```
#include <iostream>
#include <variant>
using namespace std;
enum class ErrorCode
    Ok,
    SystemError,
    IoError,
    NetworkError
};
std::variant<std::string, ErrorCode> FetchNameFromNetwork(int i)
    if (i == 0)
       return ErrorCode::SystemError;
    if (i == 1)
        return ErrorCode::NetworkError;
    return std::string("Hello World!");
}
int main()
    auto response = FetchNameFromNetwork(0);
    if (std::holds_alternative<std::string>(response))
        std::cout << std::get<std::string>(response) << "n"<<endl;</pre>
```

```
std::cout << "Error!\n";
response = FetchNameFromNetwork(10);
if (std::holds_alternative<std::string>(response))
    std::cout << std::get<std::string>(response) << "n"<<endl;
else
    std::cout << "Error!\n"<<endl;
return 0;
}</pre>
```



In the example, **ErrorCode** or a regular type is returned.

The next section discusses how to parse command line to test for the existence of parameters. Read on to find out more.