Changing the values

This lesson addresses the ways of changing values of the variant.

WE'LL COVER THE FOLLOWING

Changing Values of the variant

Changing Values of the variant

There are four ways to change the current value of the variant:

- the assignment operator
- emplace
- get and then assign a new value for the currently active type
- a visitor (you cannot change the type, but you can change the value of the current alternative)

The important part is to know that everything is type-safe and also that the object lifetime is honoured.

```
#include <iostream>
#include <variant>
using namespace std;

int main(){
    std::variant<int, float, std::string> intFloatString { "Hello" };
    cout << "We are a string: " << std::get<std::string>(intFloatString) << endl; // using get</pre>

// * assignment operator method
intFloatString = 10; // we're now an int
cout << "We are now an int: " << std::get<int>(intFloatString) << endl;

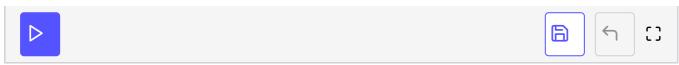
// * emplace
intFloatString.emplace<2>(std::string("Hello")); // we're now string again
cout << "We are a string again: " << std::get<std::string>(intFloatString) << endl;

// * get method
// std::get returns a reference so you can change the value:</pre>
```

```
std::get<std::string>(intFloatString) += std::string(" World");
cout << "altered string: " << std::get<std::string>(intFloatString) << endl;

intFloatString = 10.1f;
cout << "We are a float: " << std::get<float>(intFloatString) << endl;

if (auto pFloat = std::get_if<float>(&intFloatString); pFloat){
    *pFloat *= 2.0f;
    cout << *pFloat;
}</pre>
```



The next lesson will discuss more about std::variant, explaining how it
handles object lifetime.