std::string_view Creation

The different ways of initializing a string_view are listed in this lesson.

You can create a string_view in several ways:

- from `const char*` - providing a pointer to a null-terminated string
- from `const char*` with length
- by using a conversion from `std::string`
- by using `""sv` literal

Here's an example of various creation options:

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

int main() {
  const char* cstr = "Hello World";
  // the whole string:
  std::string_view sv1 { cstr };
  std::cout << sv1 << ", len: " << sv1.size() << '\n';
  // slice
  std::string_view sv2 { cstr, 5 }; // not null-terminated!
  std::cout << sv2 << ", len: " << sv2.size() << '\n';
  // from string:
  std::string str = "Hello String";
  std::string_view sv3 = str;
  std::cout << sv3 << ", len: " << sv3.size() << '\n';
  // ""sv literal
  using namespace std::literals;
  std::string_view sv4 = "Hello\0 Super World"sv;
  std::cout << sv4 << ", len: " << sv4.size() << '\n';
  std::cout << sv4.data() << " - till zero\n";
}
```

Please notice the last two lines: `sv4` contains `'\0'` in the middle, but `std::cout` can still print the whole sequence. But when you try to get the `.data()` you'll end up with string pointer so that the printing will break at the null terminator.

After creating our `string_view`, there are a lot of operations we can perform on it. We’ll discuss them next.