Problem With The Initial Example

We've learned about several risks associated with std::string_view. Let's check if they apply to the first example in this section.

The intro section showed you an example of:

```
std::string_view StartFromWord(std::string_view str, std::string_view word)
{
  return str.substr(str.find(word)); // substr creates only a new view
}
```

The code doesn't have any issues with non-null-terminated strings - as all the functions are from the `string_view` API.

However, how about temporary objects?

What will happen if you call:

```
auto str = "My Super"s;
auto sv = StartFromWord(str + " String", "Super"); // use `sv` later in the code...
```

Code like that might blow!

"Super" is a temporary `const char*` literal and it's passed as a `string_view` word into the function.

That's fine, as the temporary is guaranteed to live as long as the whole function invocation.

However, the result of string concatenation `str + " String"` is a temporary and the function returns a `string_view` of this temporary outside the call!

So the general advice in such cases is that while it's possible to return a string_view from a function, you have to be careful and be sure about the state of the underlying string.

To understand issues with temporary values, it’s good to have a look at the reference lifetime extension.