Posted by Novo on Thu, 02 Jul 2020 20:08:36 GMT
View Forum Message <> Reply to Message

Sorry for the late response.
My code is correct. Temporary String lives only during function call. This is how C++ works.
Return type is a value, not a reference. So, no temporaries ...

"The type of the ternary ?: expression is the common type of its second and third argument. If
both types are the same, you get a reference back. If they are convertable to each other, one gets
chosen and the other gets converted (promoted in this case). Since you can't return an lvalue
reference to a temporary (the converted / promoted variable), its type is a value type."

Basically, the ternary ?: is needed to convert "const char[N]" and "const char[M]" to "const char*".

On the other side, templates is a complicated thing.
If you have a non-template version of "decode" declared before template instantiation point,
compiler will choose it ...
Also MSVC is very well known for broken "two-phase name lookup". Even till these days, I believe
...
IMHO, a safer version would look like this:

```
namespace details {
 template <class T, class V>
 constexpr auto decode(const T& sel, const V& def)
 {
  return def;
 }

 template <class T, class K, class V, typename... L>
 constexpr auto decode(const T& sel, const K& k, const V& v, const L& ...args)
 {
  return sel == k ? v : details::decode(sel, args...);
 }
}

template <class T, class K, class V, typename... L>
constexpr auto decode(const T& sel, const K& k, const V& v, const L& ...args)
{
 return details::decode(sel, k, v, args...);
}
```