Subject: Re: 2023.2
Posted by koldo on Thu, 16 Nov 2023 12:03:05 GMT
View Forum Message <> Reply to Message

mirek wrote on Wed, 15 November 2023 10:41koldo wrote on Wed, 15 November 2023 08:38Hi Mirek

These sentences are very interesting.
Quote:Also, make sure that all other timer (and GUI in general) lambdas are with [=]....
You should only use [&] if you are sure that lambda is invoked immediately (e.g. with CoDo).
I almost understand why you say that but, for didactic reasons, could you explain why to include capture by value rather than by reference.

```
struct MyApp ... {
   Button btn;
   String a;
   ...
};

void MyApp::MyApp()
{
    String b;
    btn << [&] { a = b; };
}
```

This creates a dangling reference to b. In fact, if the lambda execution is not 'immediate' (before next statement basically), in most of GUI cases it makes only sense to work with member variables (e.g. 'a' in this case). But then, [=] in fact captures 'this' and that is likely MORE efficient than capturing all member variables individually.

Note that capturing this by [=] was deemed confusing by C++ comittee and they tried to improve on it with [=, this], unfortunately currently it is hard to make code compatible both ways.

Mirek

Thank you Mirek for answering this off-topic question.
Additional information about this can be found here.
I didn't know this:Quote:[=] in fact captures 'this'
In summary, especially in classes, when we sometimes put a [&], we should really have to put [this], and in some cases a [=].