Resources:

- https://stackoverflow.com/questions/18222926/why-is-list-initialization-using-curly-braces-better-than-the-alternatives
- https://isocpp.github.io/CppCoreGuidelines/CppCoreGuidelines#es23-prefer-the--initializer-syntax
- Article pointing pitfalls of uniform initialization: https://medium.com/@barryrevzin/uniform-initialization-isnt-82533d3b9c11
-

Option 1: Using = or () as much as possible

Pros:

- Usual practice of pre-C++11 old-timers coders :-)
- Most of the code base uses that because of the above reason

Cons:

- Cannot detect narrowing, like int x = somemethod_that_returns_size_t() (see https://github.com/qgis/QGIS/pull/44205/files but here that detection was seen as an annoyance rather than an opportunity, mostly because we have the issue of doing dual Qt5 & Qt6 builds)
- Prone to https://en.wikipedia.org/wiki/Most_vexing_parse issues (although rather infrequent issue)
- Caution : std::vector x{a,b} != std:vector x(a,b)

Cases where {} initialization is required as an exception to the policy:

- TBD

Option 2: Using {} as much as possible

Pros:

- Recommended (?) way / uniform initialization by C++ standard authors since C++11
- Can detect type narrowing
- Avoids most vexing parse issues (although rather infrequent issue)
- I believe can be used for class' member initialization in some situations whereas = doesn't work (example needed)

Cons:

- Older versions of cppcheck are sometimes confused by this and believe it is the beginning of a new code block (might be solved with newer ones)
- Caution : std::vector x{a,b} != std:vector x(a,b)
- Issue with QVariant ?? (not sure which one, probably that QVariant { 1 } is interpreted as QVariant (const QList<QVariant>{1}) ??)

Cases where = or () initialization is required as an exception to the policy:

- TBD