

# KDE's Journey to Qt6

### and Beyond

2023/11/29, Qt World Summit Nicolas Fella – nicolas.fella@kde.org













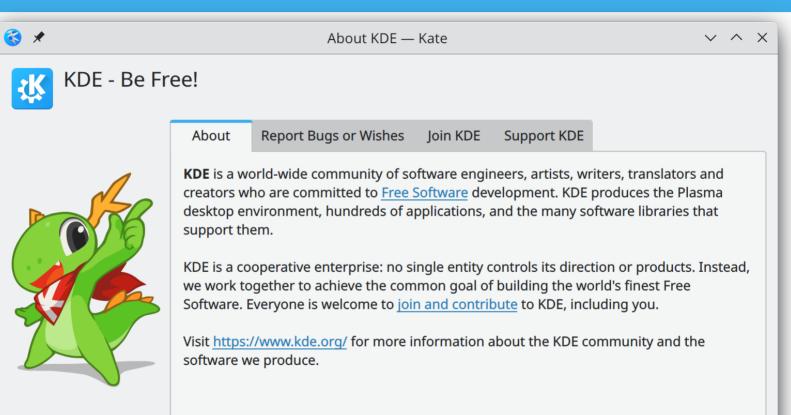
















- KDE Contributor since 2017
- KDE Software Platform Engineer since 2023
- Leading the Qt6 transition in KDE
- Software Engineer at KDAB
- Qt contributor





- betterCode Qt 2021: KDEs Weg zu Qt6
- Qt DevCon 2022: KDE's journey to Qt 6
- QtCon Brasil 2022: KDE's journey to Qt 6 and beyond
- Various talks at KDE's Akademy conference



- KDE has *a lot* of code: ~15 Mio. LoC
- Virtually all C++, Qt5, QML
- Many independent subprojects
- Multi-generational code
- Both Widgets and QML
- Graphics-intensive applications
- CMake build system



- 80+ modular libraries, solving recurring problems
- Available for external users
- Stable API + ABI
- Qt6 means new major versions for KDE Frameworks



- First planning session at Akademy 2019
- Setting initial design goals and principles
  - Easy porting without surprises
  - Improve APIs and drop old ballast
  - Separate Core/Widgets/QML
  - Improve cross-platform support



- Preparation work in 2019 and 2020
- First proper Qt6 builds in 2021
- First Qt6-only builds in 2023
- Qt6-based alpha releases in November 2023
- Release planned for February 2024



# What took you so long?



- Some things just take time to port
  - QRegExp
  - QStringRef
  - QDesktopWidget
  - Shader
  - Third-party dependencies



- Some functionality was/is missing:
  - QTextCodec
  - QtGraphicalEffects
  - QtX11Extras
  - QML TreeView



- Porting QML code was easy and hard at the same time
- Easy because of few breaking changes
- Hard because of no good way to keep Qt5 support
- Wish: #ifdef version check in QML



- Naturally we found some Qt bugs
- Many of them reported and/or fixed by us
- Contributing upstream is vital for KDE
- Lack of 5.15 LTS is a challenge



- An openly maintained Qt
- Much improved Wayland support
- Many small but useful API additions
- Improved QML tooling
- Performance improvements



#### KDE For All Sustainable Software Automation

Improve accessibility across our products Reduce energy consumption and hardware waste Automate and systematize internal processes



- Formalize and automate internal processes: code review, release management, etc
- Tools like qmlformat and qmllint can help
- Qt and KDE can learn from each other



- Reduce energy consumption of software
- Avoid hardware obsolescence
- Performance improvements in Qt help



- Improve accessibility of our products
- Lots of activity in Qt
- Still some open issues, esp. in QtQuick



## Strategic Challenges



- QML is evolving rapidly
- We have not fully caught up
  - qt\_add\_qml\_module not usable for libraries
  - Lots of "legacy" QML code
  - Bespoke QML deployment
  - Issues with third-party QML modules and tooling



- Future of the Linux desktop
- Important target platform for Qt
- Substantial contributions from KDE
- More work needed in Qt and upstream



- Widgets vs QtQuick is hotly debated in KDE
- QtQuick is lacking on desktop platforms:
  - No native dialogs, popups, menus
  - Keyboard navigation is hard
  - Lack of high-level controls
  - Different theming system than widgets
  - No unified action class
  - Developer Experience



- KDE's Qt6 transition is drawing to a close
- Quite smooth, but not without issues
- Lots of modernization work ahead
- Some strategic issues still exist
- Qt6 helps KDE achieve its goals

## Qt6 will has fixed it!



### Questions?