Modern Application Development

for Any Domain

6/16/2021

SOLUTIONS ENGINEER

Corey Pendleton

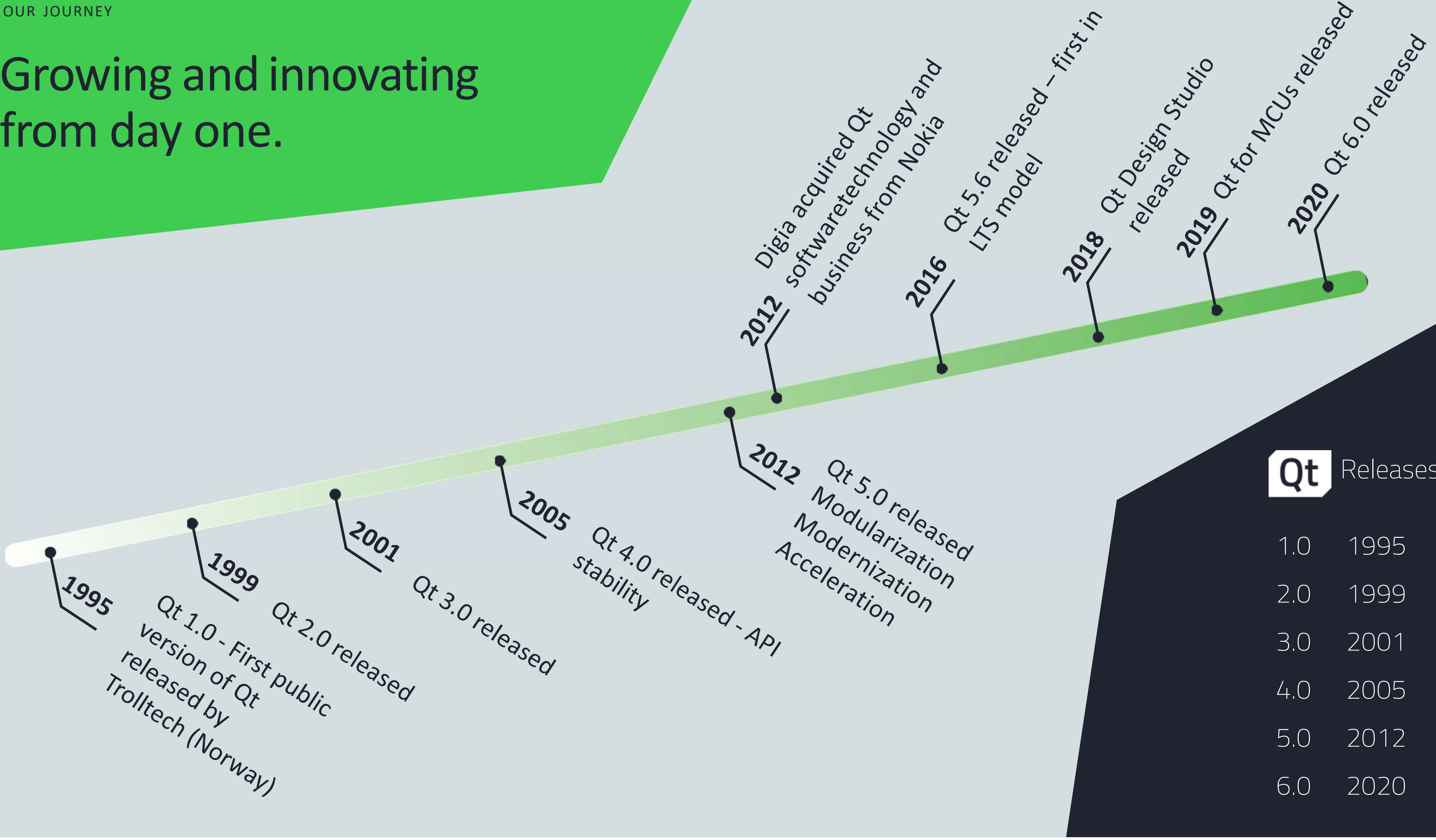
- **Software Engineer**
  - 15 years in automation and consumer devices
  - 7 years using Qt framework
  - Focus on front-end HMIs
- **Solutions Engineer**
  - Sales Support
  - Development Workflow
  - Qt for MCUs Development

Agenda

1. Meet Qt
2. Qt Framework
3. Demos
4. Qt 6 Roadmap

OUR JOURNEY

Growing and innovating from day one.



1995 Qt 1.0 - First public version of Qt released by Trolltech (Norway)

1999 Qt 2.0 released

2001 Qt 3.0 released

2005 Qt 4.0 released - API stability

2012 Qt 5.0 released Modularization Modernization Acceleration

2012 Digia acquired Qt softwaretechnology and business from Nokia

2016 Qt 5.6 released – first in LTS model

2018 Qt Design Studio released

2019 Qt for MCUs released

2020 Qt 6.0 released

Qt Releases

1.0	1995
2.0	1999
3.0	2001
4.0	2005
5.0	2012
6.0	2020

QT FRAMEWORK

More Than a Collection of Libraries

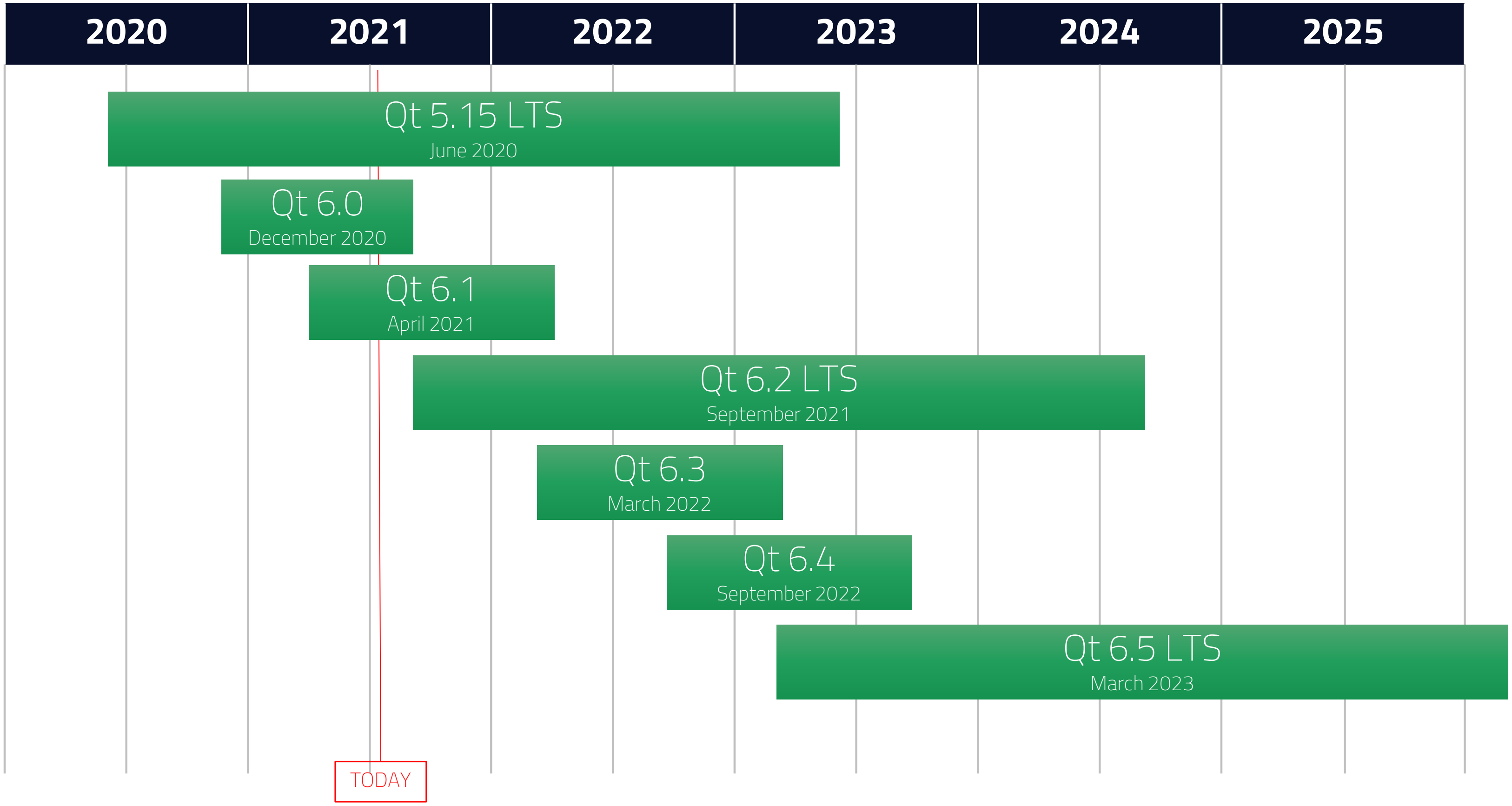
- **Frameworks are opinionated**
  - Consistent APIs and documentation
  - Structure
  - Best practices – Frameworks provide proven solutions
  - Dictates how to do things – can be extended
- **Frameworks come with a toolbox**
  - IDE, toolchains, etc.
  - Makes it easy to apply best-practices
- **A good framework drives structure and consistency**

"Collection of libraries"

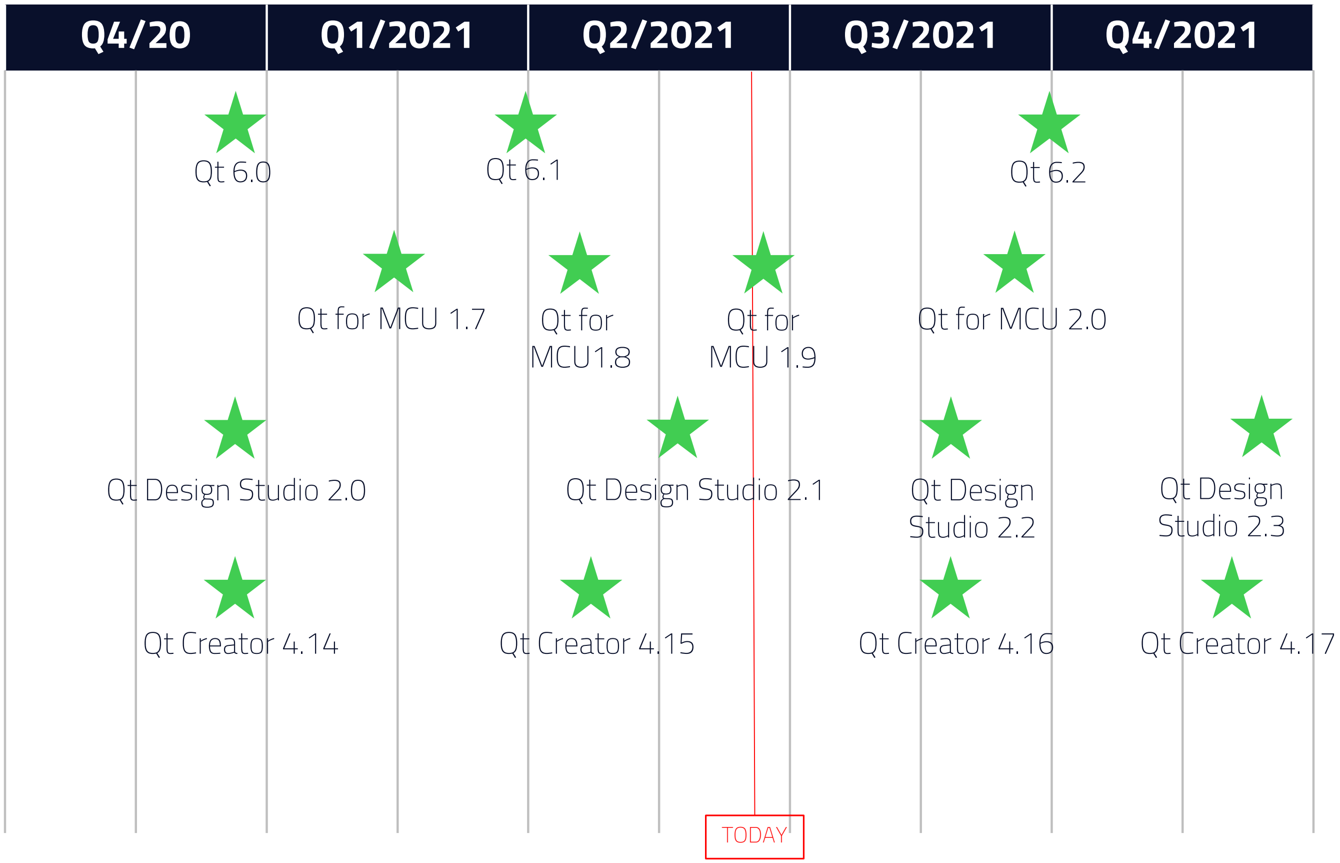
VS

Framework

Qt 6 Roadmap


2020
2021
2022
2023
2024
2025
Qt 5.15 LTS
June 2020
Qt 6.0
December 2020
Qt 6.1
April 2021
Qt 6.2 LTS
September 2021
Qt 6.3
March 2022
Qt 6.4
September 2022
Qt 6.5 LTS
March 2023
TODAY

Timeline 2021 – All products



Q4/20

Q1/2021

Q2/2021

Q3/2021

Q4/2021

Qt 6.0

Qt 6.1

Qt 6.2

Qt for MCU 1.7

Qt for MCU1.8

Qt for MCU 1.9

Qt for MCU 2.0

Qt Design Studio 2.0

Qt Design Studio 2.1

Qt Design Studio 2.2

Qt Design Studio 2.3

Qt Creator 4.14

Qt Creator 4.15

Qt Creator 4.16

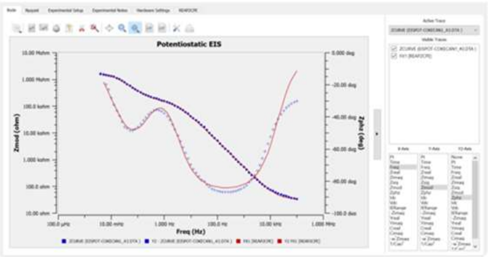
Qt Creator 4.17

TODAY

Qt Framework

QT FRAMEWORK DETAILS

Optimal UI solutions for each use case

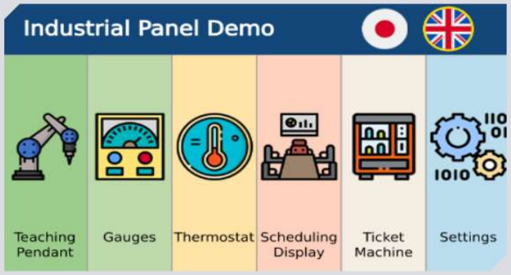


2D/ 3D UIs

Qt Quick declarative UI design (QML) for fluid, modern touch-based User Experiences

Web / Hybrid

Use HTML5 for dynamic web documents, Qt Quick for native interaction



Remote UIs

Run headless device UIs remotely in the browser using WebGL or WebAssembly

Qt Widgets

Customizable C++ UI controls for traditional desktop look-and-feel or more static embedded UIs for more limited devices

QT FRAMEWORK DETAILS

Widgets

Easy to use, easy to extend, easy to style

Most suitable for desktop

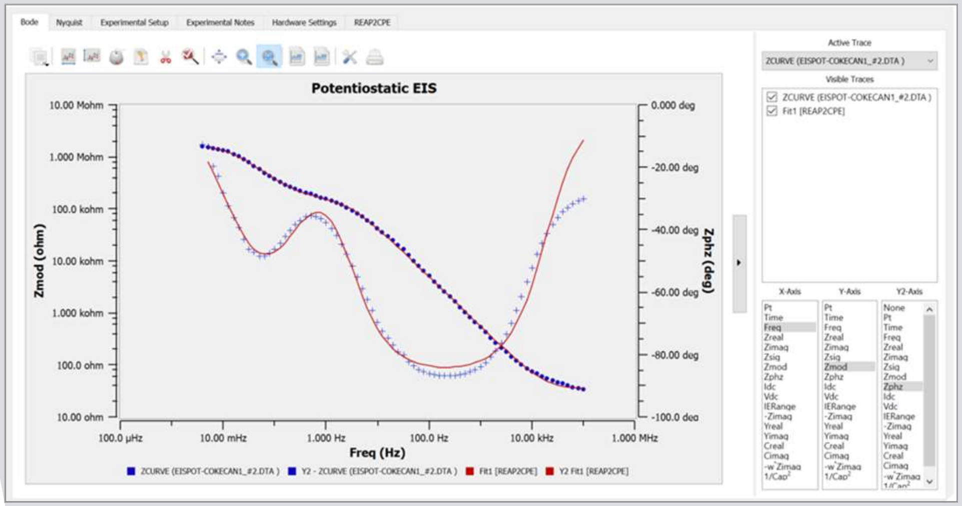
Native desktop look'n'feel – easily stylable

Easy to scale to any display size and orientation

WYSISYG UI design tool

- Create the UI sketch with a custom style in minutes
- Plenty of controls available: buttons, sliders, LCD number, tree view, dock widget

No graphics processor needed => extends the HW base



Qt and C++

QT FRAMEWORK DETAILS

2D / 3D UIs with QML

Nice modern, phone like **UIs for all targets**

- Especially for embedded and mobile

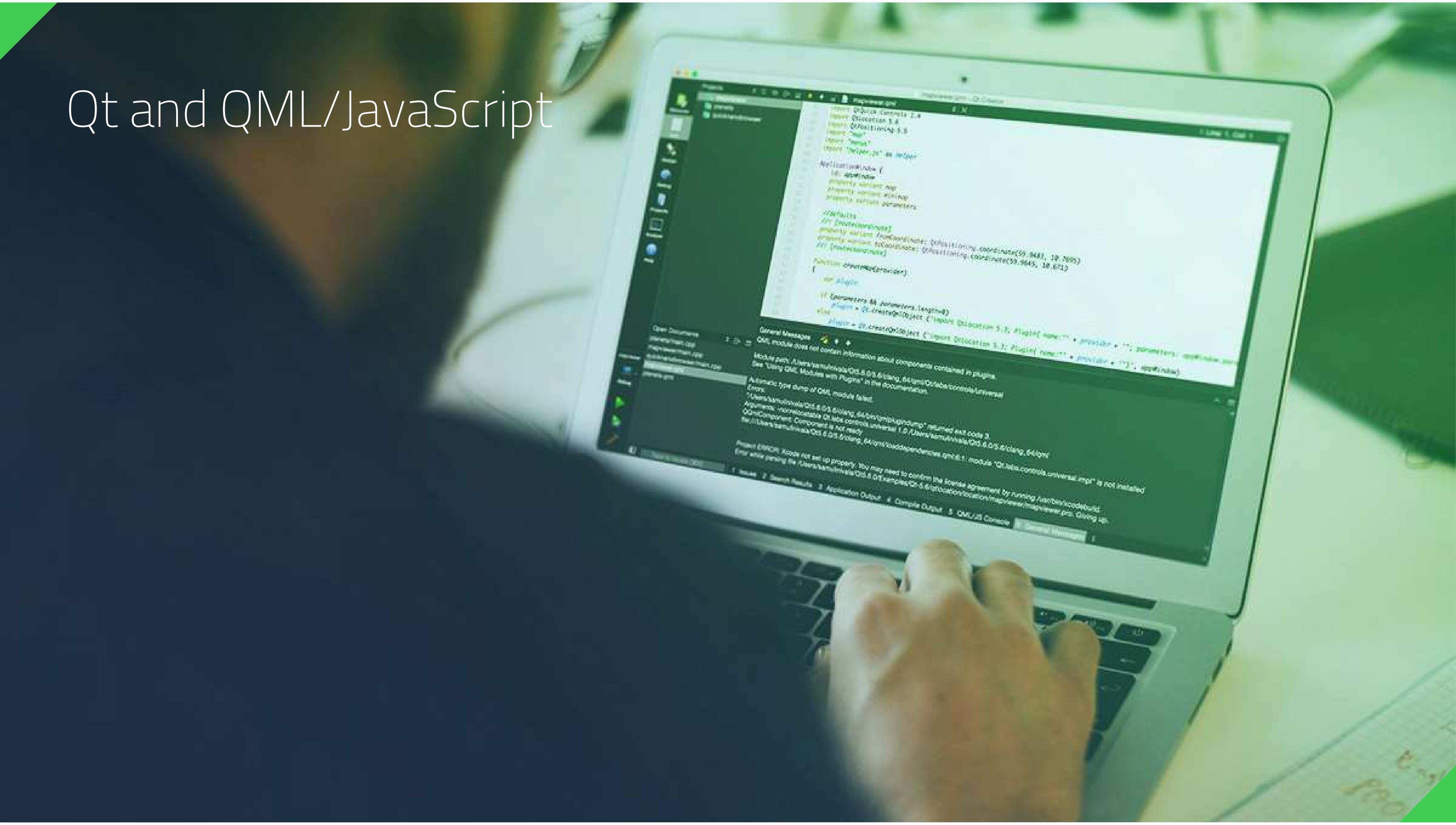
WYSISYG UI design tool - **Qt Design Studio**

- Generates UI implementation in QML

QML declarative language for creating UIs

- Easy to learn
- Quick to prototype even in the target HW – no compilation needed
- Can be compiled to the native code to get the best possible performance
- Great tooling to find rendering bottlenecks
- HW accelerated on targets with the GPU

Qt and QML/JavaScript



QT FOR PYTHON

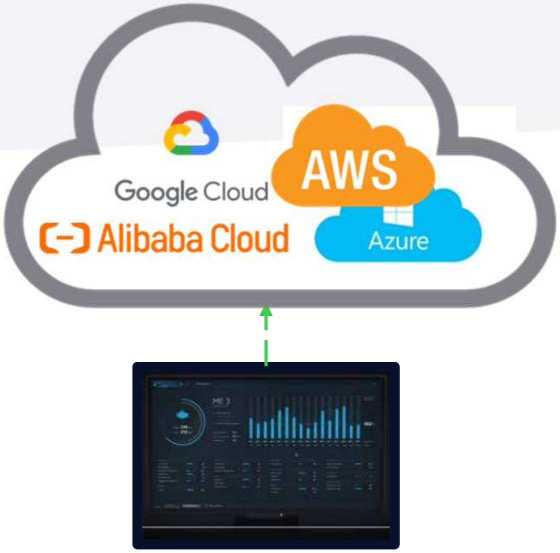
Qt for Python (PYSIDE)

- **Easy to extend applications**
  - Cross platform plugins without recompilation
  - Scripting support
  - Full control of runtime environment
- **Integrate Machine Learning**
  - Python most used language for ML
  - Easily accessible via Qt for Python
- **Remove programming language barrier**
  - 4th most popular language (StackOverflow Insights 2019)
  - 2nd most loved language
  - 1st most wanted language

QT FRAMEWORK DETAILS

Connectivity

```
{
  “id”: “961b276c-40f7-11ea”,
  “location”: “b77f-2e728ce88125”,
  “rpm”: 6200,
  “temp”: 27.4,
  …
}
```



Attach to peripherals

Control external hardware via any protocol.
CANbus, Modbus, Serial Port, Bluetooth, BTLE,…

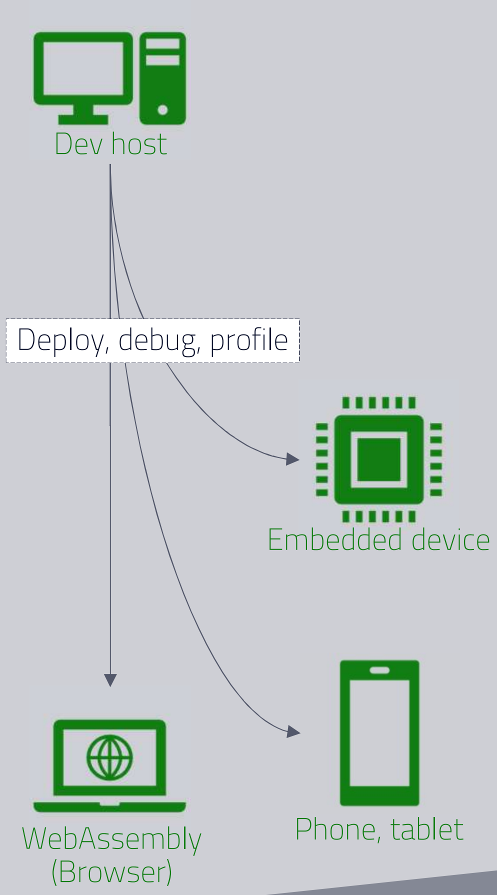
Data Serialization

Store and export data to industry standard formats.
JSON, CBOR, XML,…

Cloud synchronization

Publish telemetry data, visualize health status, database storage.
Protocol layer: MQTT, CoAP, OpcUA, KNX, HTTP,…
Transport layer: TCP, UDP, Websockets, Local sockets,…

Platforms


Dev host
Deploy, debug, profile
Embedded device
WebAssembly (Browser)
Phone, tablet

CELOS

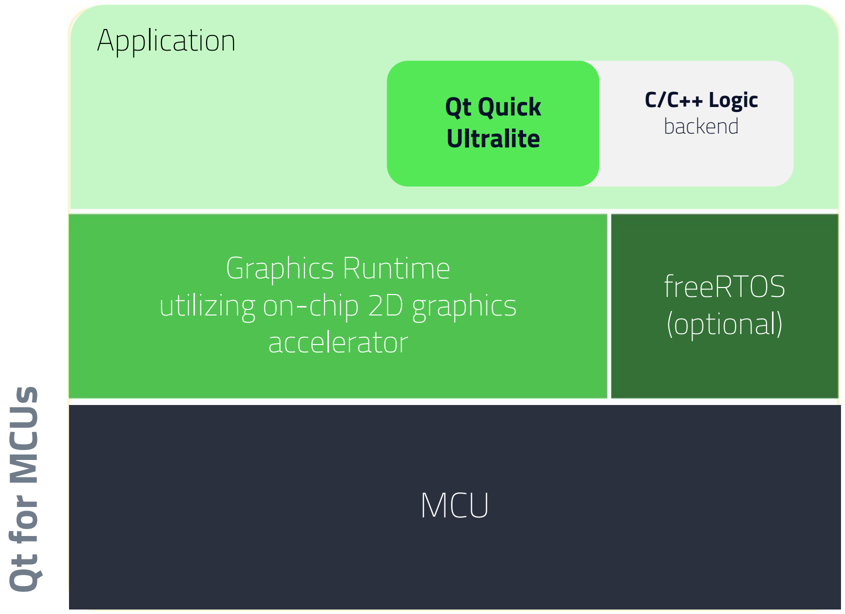
Cooling System Data
67°C

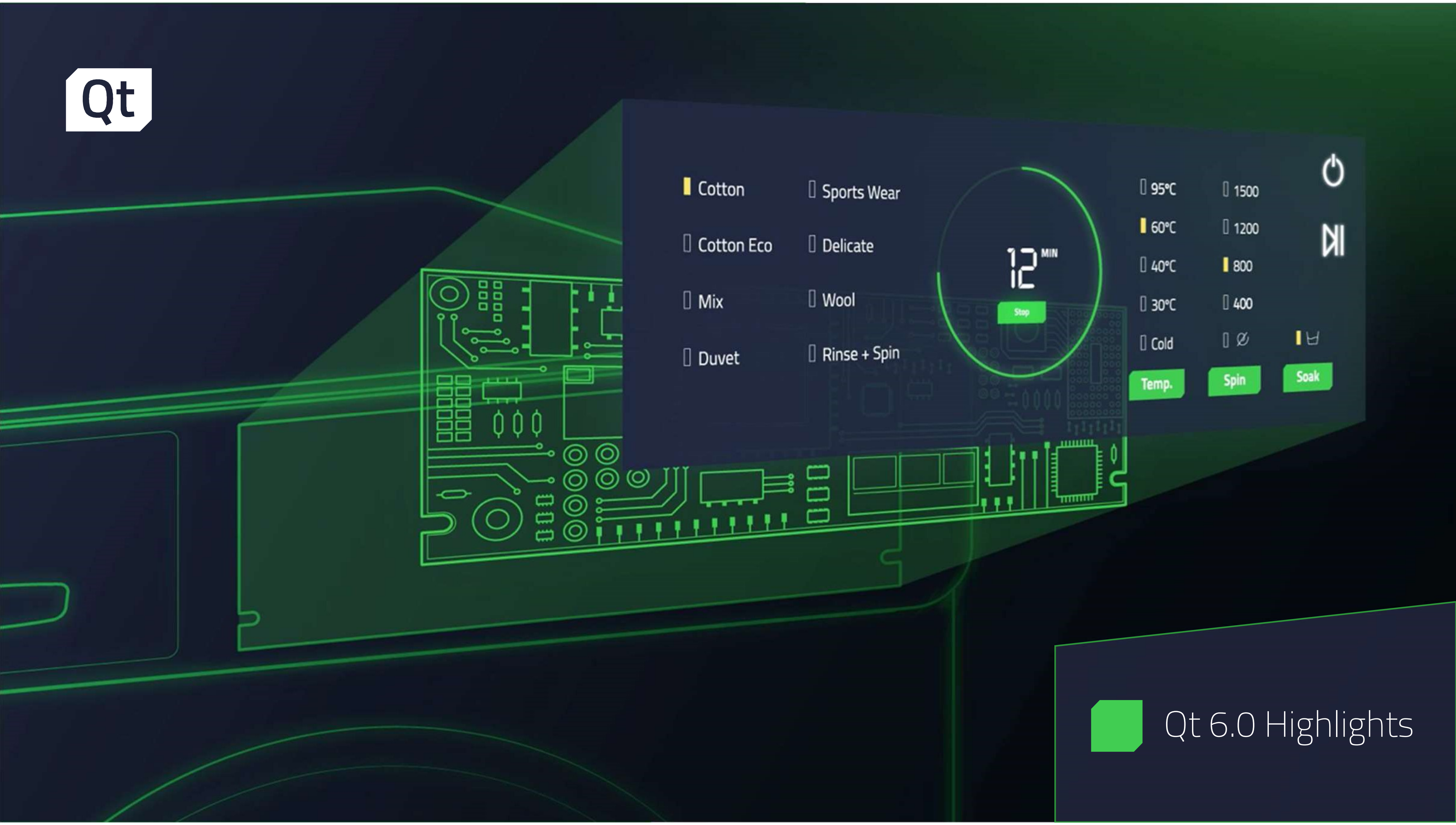
Qt for MCU

TARGETS

Qt on microcontroller hardware

- Declarative UI and OOP – the best of both worlds
  - Re-use and deploy same QML based UI while implementing Application logic in standard C/C++
- Ultimate Performance. Tiny Footprint.
  - A new rendering engine uses HW 2D accelerators to achieve good graphical performance. The runtime itself has a very small footprint (starting from ~80KB)
- Supports on a wide range of MCUs and RTOSs
  - MCUs from ST, NXP, Renesas, Cypress/Infineon, Xilinx UltraScale+
  - Bare Metal or FreeRTOS (on selected boards)





Qt 6.0 Highlights

Ecosystem

C++ 17

- Update to latest standards
- Improved code readablility
- Better performance
- Easier to maintain

CMAKE

- Industry standard build system
  - Now used to build Qt as well
  - Harmonized environment, no more custom tools
- Wide feature set
- Large developer ecosystem
- QMake still supported for Qt-based projects

RENDERING HARDWARE INTERFACE

Qt RHI

UX

Future

Scale

- Create hardware-accelerated user-interfaces on any rendering platform
  - OpenGL, Vulkan, Direct 3D, Metal
- New Qt Shader Tools
  - Write rendering code once, deploy to any hardware
- Add new hardware targets in no time

```
class ImageWidget : public QWidget
{
    Q_OBJECT
public:
    ImageWidget(QWidget *parent = 0);
    void openDirectory(const QString &path);
    void grabGestures(const QList<Qt::GestureType> &gestures);
protected:
    bool event(QEvent *event) override;
```

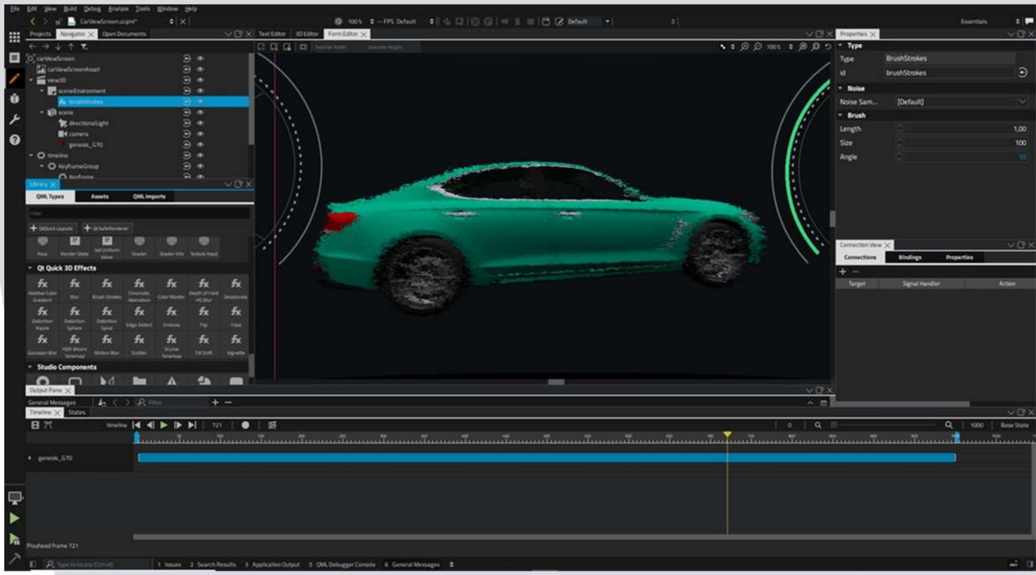
⚠ zero as null pointer constan

NEXT-GEN HMI

Qt Quick 3D

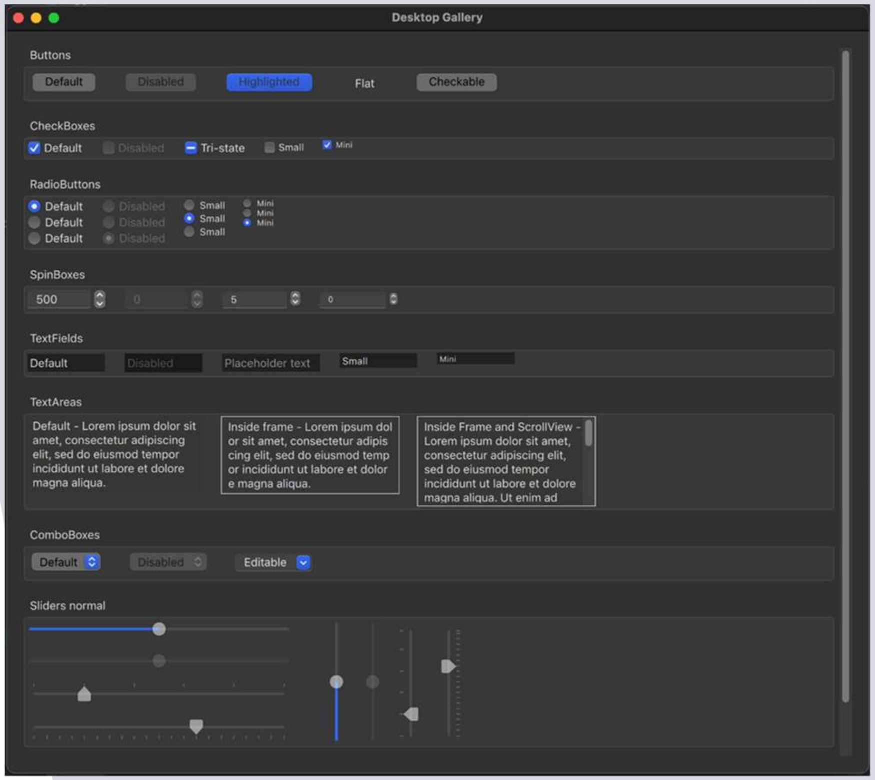
UX | Future | Scale

- Merge 2D and 3D content
  - One technology stack instead of two concurrent requiring synchronization
- Many improvements and new features



NATIVE LOOK & FEEL

Qt Quick Controls 2 Desktop Styling


Desktop Gallery
Buttons
Default
Disabled
Highlighted
Flat
Checkable
CheckBoxes
Default
Disabled
Tri-state
Small
Mini
RadioButtons
Default
Default
Default
Disabled
Disabled
Disabled
Small
Small
Small
Mini
Mini
Mini
SpinBoxes
500
0
5
0
TextFields
Default
Disabled
Placeholder text
Small
Mini
TextAreas
Default - Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.
Inside frame - Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.
Inside Frame and ScrollView - Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad
ComboBoxes
Default
Disabled
Editable
Sliders normal

- Pixel-perfect, native looking controls
- Seamlessly integrate into underlying OS
- Streamlined behavior in your UIs

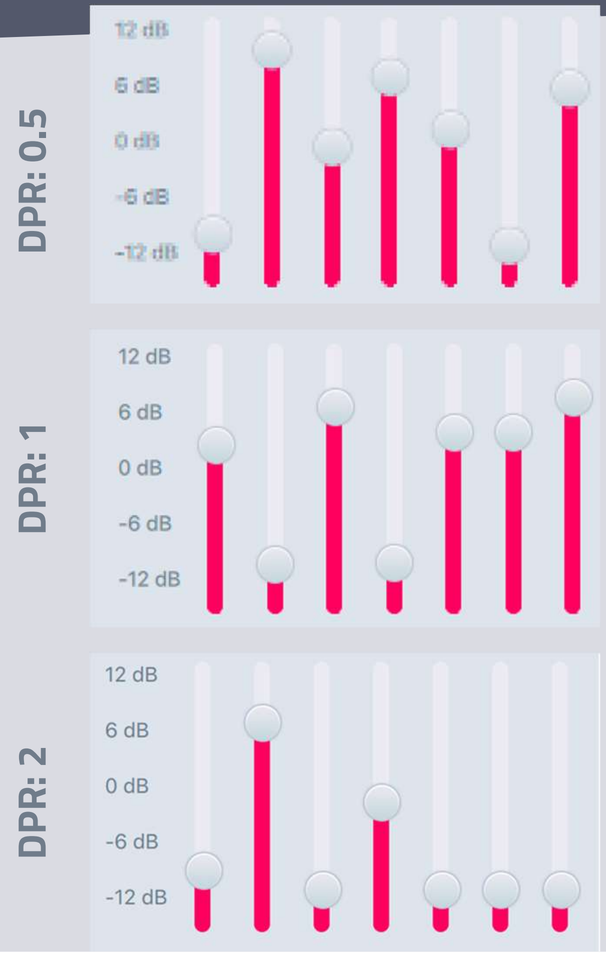
NATIVE LOOK & FEEL

HiDPI support

UX

Future

Scale



- New: Fractal scaling
  - Allows for monitors with e.g. 125% setup
- Settings per display
- Forward size calculations to Qt

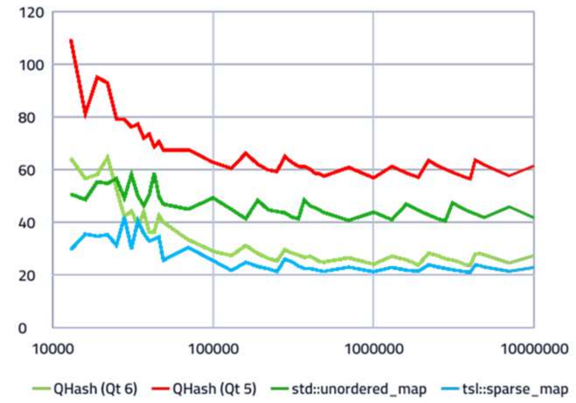
HIGH PERFORMANCE, LOW MEMORY

New QProperty System

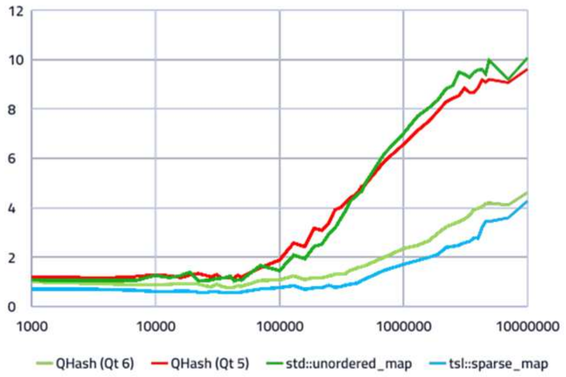
UX Future Scale

- Binding support in C++
  - Bring the best part of QML to Qt
  - Seamless integration with QObject
- Lazy evaluation
  - Spare non-required calculations
  - Much faster code
- Compatible with Qt5
  - Source compatible
  - Only port where needed

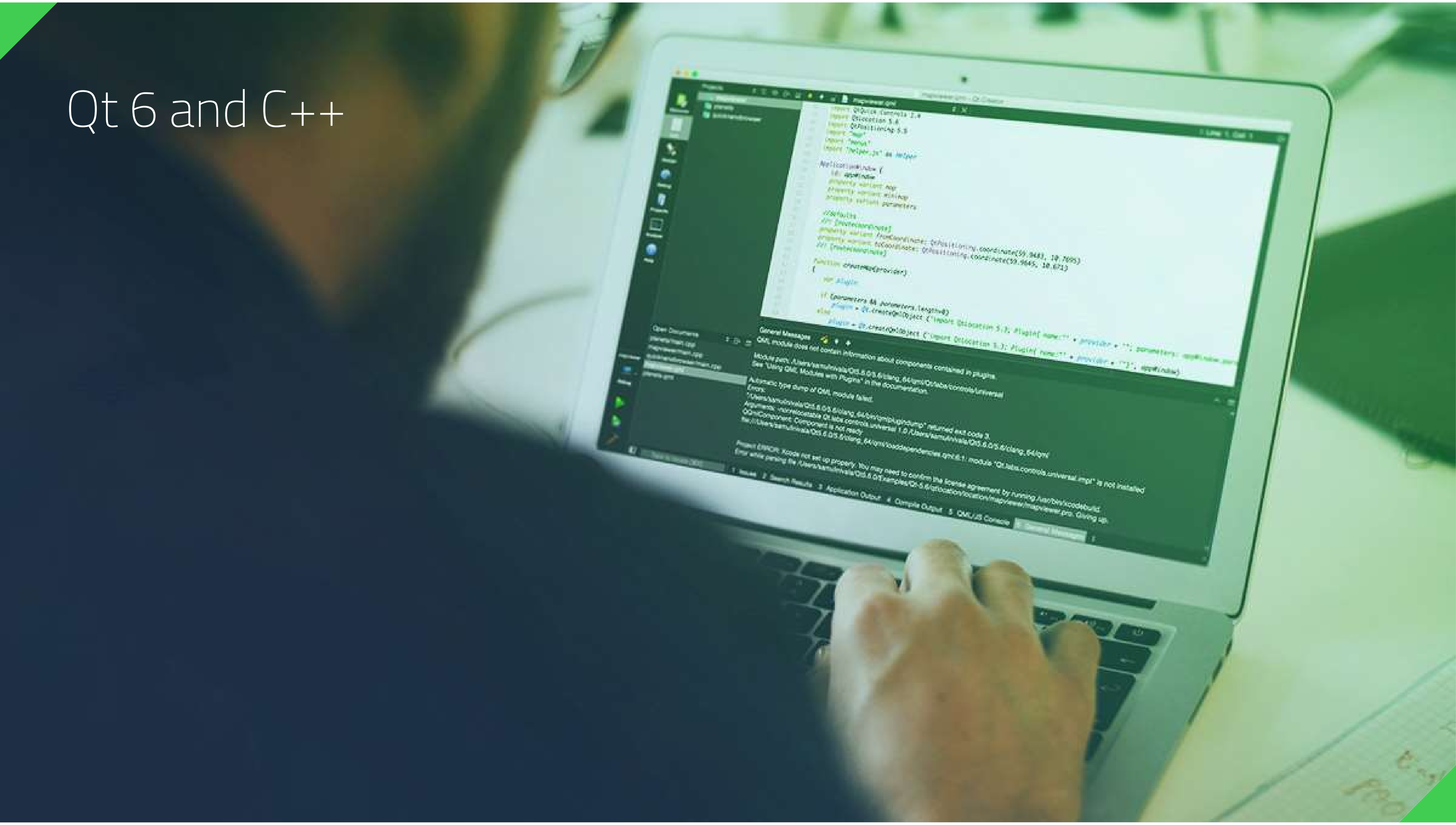
Memory usage *(bytes/entry)*



Benchmark results *(lower is better)*



Qt 6 and C++



Thank you!

Corey Pendleton – Solutions Engineer
corey.pendleton@qt.io

Amir Alvarez – Account Manager
amir.alvarez@qt.io