

# QT AND THE WEB

Where are we today?

Qt World Summit Berlin 2019

# About basysKom

## basysKom GmbH

- Is a Qt Service Partner since 2004
- Is located in Darmstadt and Nürnberg
- Is employing ~30 people
- Is part of the UX Gruppe
- Provides services (consulting, training, coaching and development) around Qt
- Focuses on technical/industrial applications of Qt (HMI and application development)

## About myself

- Development Lead @ basysKom

# Introduction

The web is now around 30 years old

Hypertext, HTML, HTTP(S), web browser, web server, (...) are omnipresent

The web with its immense success/massif adoption had a strong influence on technology

For a lot of use-cases web-technologies are the default

# Is the web the Holy Grail?

## Problems with web technologies

- Strongly fragmented ecosystem
- Lots of churn (short-lived APIs, fashion-driven)
- Offline is a special case
- Limited use on (low-end) embedded hardware
  - System integration can be a mess
  - Performance issues

## It's a web's world

So even in 2019 there are good reasons to develop native applications

**Still a native framework has to prove itself capable in a „web world“**

- Consuming web services (Qt as a client)
- Providing web services (Qt as a server)
- Rendering web content with Qt
- Getting Qt into the browser

# QT AS A CLIENT



## Use-Cases: Qt as a client

### Task:

- Interface web-focused services

### Examples

- Upload/download of files
- Interacting with REST- and/or WebSocket-APIs

### Things to consider

- Serialisation of API data
- Transport security
- Authentication/authorization

# HTTP(s)-client

## QNetworkAccessManager

- C++ API
- Part of QtNetwork
- A bit of an abstract name for a HttpClient? ;-)
- API can feel a bit dated when coming from more modern asynchronous environments
  - In Qt since 4.4

## XmlHttpRequest

- JS-API that can be used from QML
- Mostly analog to the browser API of the same name
- Can be used to pull data into a QML-only/-focus Qt Quick application

**Both support SSL/TLS (via OpenSSL)**

# WebSocket-client

**WebSockets == bidirectional connection between client and server**

- Enables the server to push events into the client (browser)

## WebSockets in Qt

- QWebSocket provides a client
- Part of the QWebSockets module
- Provides a C++ and QML API
- Supports TLS (wss://)

# Serialization/Deserialization

## JSON

- Typical format used on Web-APIs
- Supported through QJsonDocument and friends
- Part of QtCore
- No JSON-Schema support (there are add-ons that work with QJsonDocument though)

## XML

- Old-school, but still around

### Various options in Qt

- QXMLStream{Reader|Writer} in Qt Core
- QDomDocument in Qt XML
- Qt XML Patterns for a validating parser

# Authentication/Authorization

**Clients need to authenticate themselves against (web) services**

- Ideally without giving the client the username/password
- Ideally without giving the client wholesales access

**Token-based authentication/authorization**

**Supported in Qt through the QNetworkAuthorization module**

- OAuth1, 2

# QT AS A SERVER



# Use-Case: Qt as a server

## Task

- Provide REST and/or WebSocket-APIs with Qt

## Examples

- Interfaces for test-automation
- Headless Qt (Qt as a friendly/better C++ to implement an embedded firmware)
- Integration point for other applications (e.g. order import)
- Integration of legacy applications

## Use-Case: Qt as a server

### Solution: QHttpServer!

- A Qt component that has been missed by a lot of people
- There are quite a few Qt-ish HttpServer projects on github, none that has critical mass is properly maintained

### Solution: QWebSocketServer

- Part of the Qt WebSocket module
- Available since Qt 5.4
- C++ and QML bindings

### Now there is an „official“ take by The Qt Company

- <https://code.qt.io/cgit/qt-labs/qthttpserver.git/>
- Currently still a labs project!
- GPL + commercial licensing

# RENDERING WEB CONTENT WITH QT



# Use-case: embedding web content

## Task

- Existing web content needs to be rendered within a Qt application
- Examples
  - Generic web browser
  - Walled gardens
  - Games
  - Reports
  - Output of authoring tools that produce web content

## Solution

- QWebEngine
  - Embeddable browser component based on Chromium
  - QML & C++ APIs
  - Complex API – enables complex applications
- QWebView
  - Wraps a native browser component on Android/iOS
  - Falls back to QWebEngine on other platforms
  - QML only
  - Simple API – high level integration

# Use-case: Web-HMI & Qt/C++ backend

## Task

- A mixed mode application where the frontend is done with web-technologies and the backend with Qt/C++
- Deeper integration compared to a classic client/server scenario
- Motivation
  - Performance
  - Legacy code (aka. „valuable business logic“)

## Solution

- QWebEngine to „host“ the HMI
- QWebChannel to enable access from browser-side JS to QObjects
- See: [doc.qt.io/qt-5/qwebchannel.html](http://doc.qt.io/qt-5/qwebchannel.html)

# Use-cases: embedded web browser

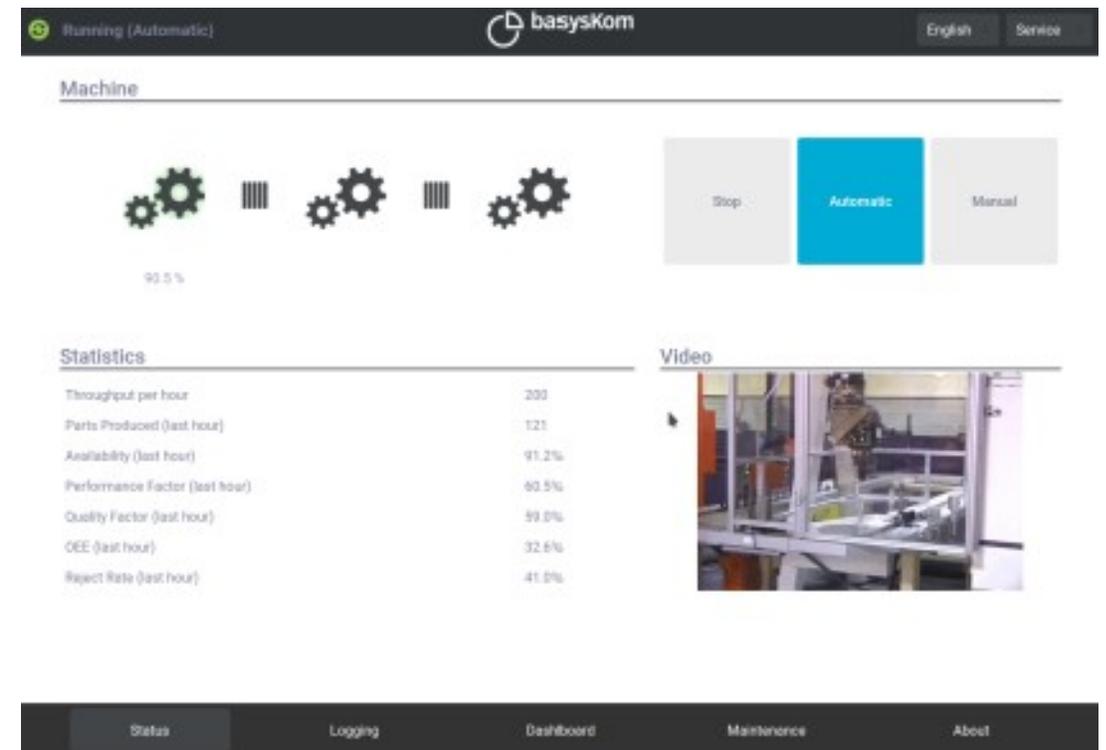
## Task

- Web browser for an embedded system
- Full-screen, no Window-Manager
- Not actually a „Qt application“, just taking advantage of the pre-existing system integration work

## Solution: Qt WebEngine (again)

## Caveats

- System-Integration can be tricky
- Large component
- Requires more powerful hardware (compared to Qt Quick)



The screenshot displays the basysKom embedded web browser interface. At the top, a status bar shows 'Running (Automatic)' and the basysKom logo. Below this, the 'Machine' section features three gear icons and a '90.5%' indicator, with control buttons for 'Stop', 'Automatic' (highlighted), and 'Manual'. The 'Statistics' section contains a table of performance metrics:

Statistics	
Throughput per hour	200
Parts Produced (last hour)	121
Availability (last hour)	91.2%
Performance Factor (last hour)	60.5%
Quality Factor (last hour)	99.0%
OEE (last hour)	32.6%
Reject Rate (last hour)	41.0%

To the right of the statistics is a 'Video' section showing a live feed of a robotic arm in a factory setting. At the bottom, a navigation bar includes links for 'Status', 'Logging', 'Dashboard', 'Maintenance', and 'About'.

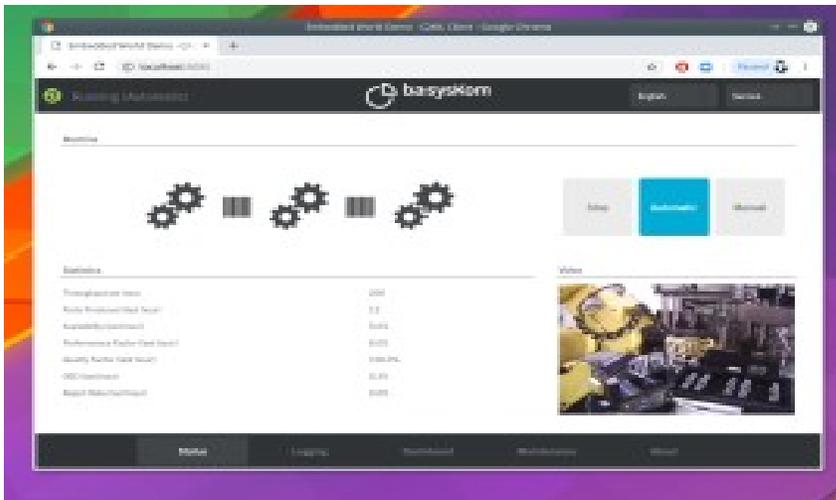
# GETTING QT INTO THE BROWSER



# Use-case: the browser as a remote display

## Task

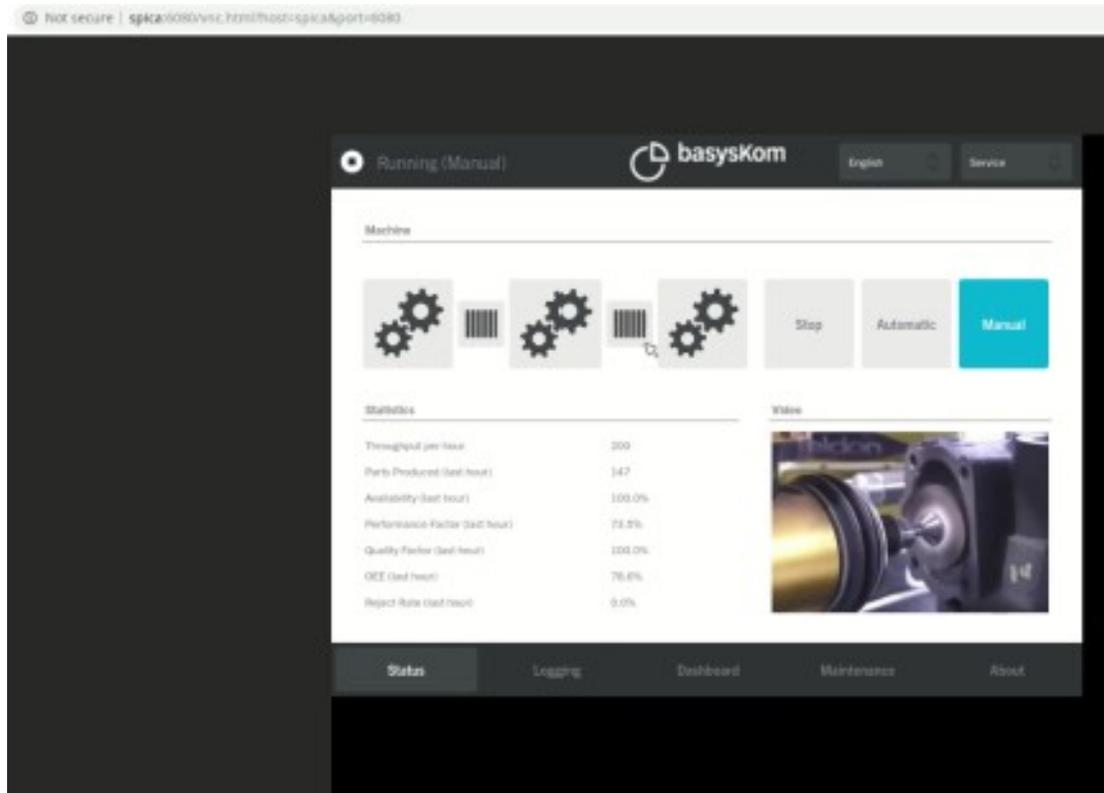
- An HMI/application that is written in Qt should render into a browser
  - Device without a display
  - Remote access to a device (which has a display)



## Solution #1: WebGL streaming

- A Qt QPA that is streaming webgl commands
  - `myApp -platform webgl`
- Qt Quick only
- GPL + Commercial licensing
- Resolution independent
- Latency is critical
- One user at a time
- Example: <https://www.qt.io/bosch-built-with-qt>

## Use-case: the browser as a remote display



### Solution #2: VNC

- VNC QPA + Websocket-Proxy + NoVNC (JS-Client)
- QWidget & Qt Quick (when using the sw-renderer)
- Is sending bitmaps
- Resolution is fixed!
- Latency is critical
- Also one user at a time

# Use-case: running Qt in the browser

## Task

- An application written in Qt needs to run inside an unmodified web browser
- Nothing is to be installed

## Examples

- Headless devices
- Inhouse software distribution

## Excursion: WebAssembly

- Portable runtime (byte code, APIs)
- Implemented by all major browsers
- Security wise the runtime has the same limitations as a regular website

## Use-case: running Qt in the browser

### Solution: Qt for WebAssembly

- Official platform with Qt 5.13
- [www.qt.io/qt-examples-for-webassembly](http://www.qt.io/qt-examples-for-webassembly)

**Relatively large downloads (~20MB for a Qt Quick application)**

**Link time is slow...**

**Application needs to deal with browser limitations**

- File access
- Network access

# WRAP UP



## Wrap up

### Four perspectives on Qt and the web

- Client
- Server
- Rendering
- Qt in the browser

### More?

- ADL, Swagger, openAPI
- Templating, HTML generation
- Complete WebApplications in Qt?
- ...

### Come talk to us about your „Qt and the web“ projects

- You can find us at the basysKom booth

# QUESTIONS

**Frank Meerkötter**

Development Lead

[frank.meerkoetter@basyskom.com](mailto:frank.meerkoetter@basyskom.com)