Qt Competitive Sales Pitches

How to compare Qt to some of the other technologies that are out there

Confidential!
Internal use only

In Progress - Draft

20.04.2017

Introduction

› This is an internal Qt Company presentation. Do not share this deck with prospects.

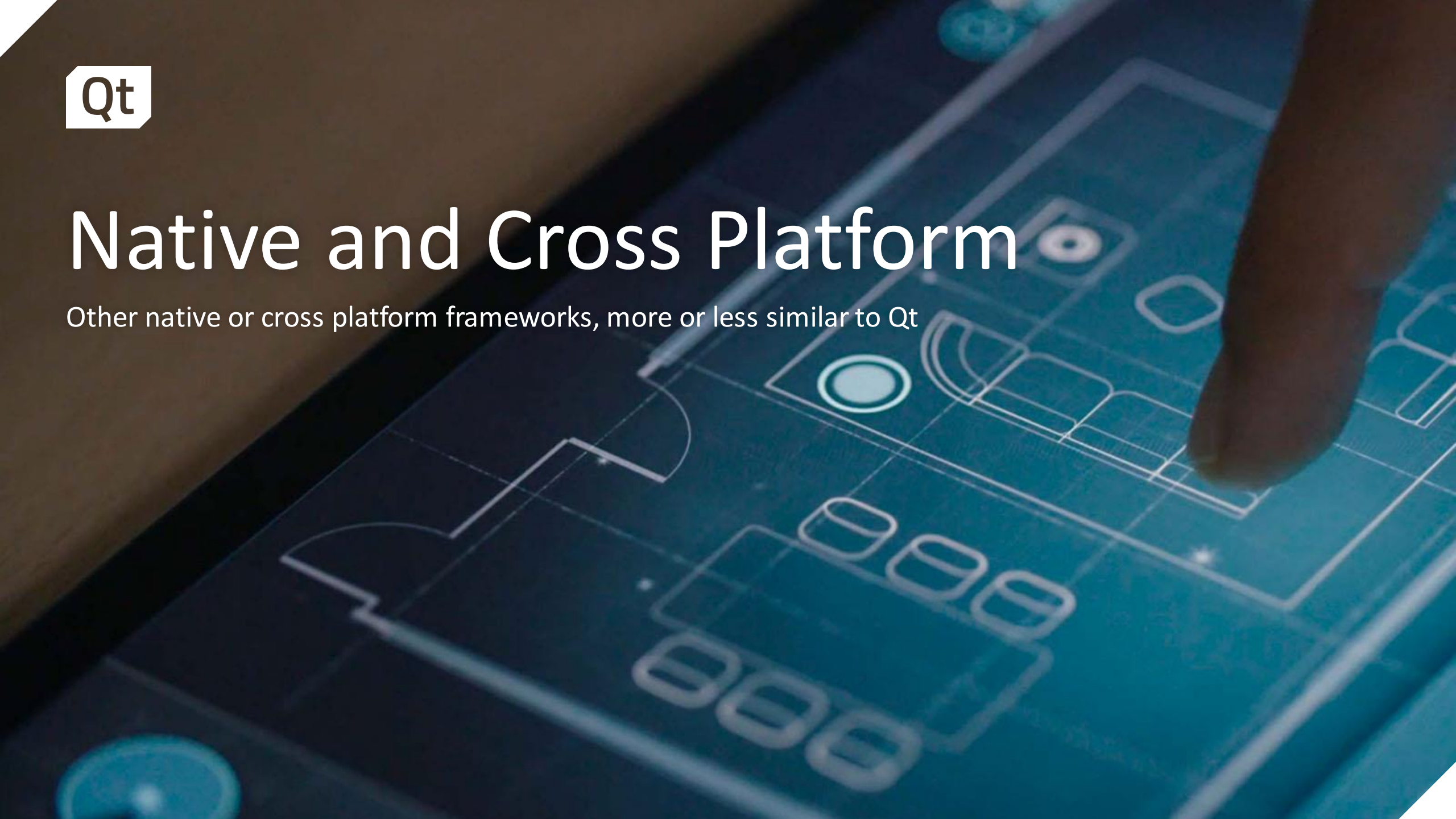
› The goal of this presentation is to equip anyone to talk to the major differences between Qt and various other technologies.

› We want to show the advantages of Qt in many different scenarios

› Our point of view might be somewhat biased, but this is an honest reflection of the market and our own perception of these technologies.

› Make sure you listen to the customer! People have different challenges, and need different pitches!

› For further questions or details, please contact your Pre-Sales Engineer, or Qt Product Management



Native and Cross Platform

Other native or cross platform frameworks, more or less similar to Qt

Competition Overview

Qt is behind | Same-isch | Qt is better | Qt is superior

	User Experience	UI Design Experience	SW Developer Experience	Cross Platform functionality	Performance	Footprint	Quality	SDK	Overall Offering
Xamarin									Mobile
Crank									UI only
Kanzi									Auto UI
Java									Bad API's
Embarcadero Delphi									
Enlightenment Toolkit (EFL)									Extremely hard to use
wxWidget									Old school
GTK+									Very limited
Android					N/A	N/A	N/A		Google lock-in
Altia									Medical, Auto

Xamarin

	Status	Stronger than Qt	Benefit of Qt	How to address this competitor
User Experience		Simpler to make mobile apps than it is with Qt	C++ is better than C#. Qt is a full blown	Xamarin is a much thinner UI layer, while Qt is a complete application development framework.
UI Design Experience		Wider mobile OS feature support	The UI implementation needs to be specific for each OS	Qt delivers pixel perfect UI across any device. Rely only on the QML, so you are in control. This is a different approach than Xamarin's native UI
SW Developer Experience		Full API support for iOS and Android	Xamarins Uses different API's on the OS'es. Qt is a unified API	
Cross Platform		Provides full API support for all the mobile systems.	Not really cross platform. Different implementations needed	
Performance		Comparable		
Footprint				?
Quality		Looks good.	Looks awesome!	
SDK		Tight integration with Visual Studio	Qt Creator is better for embedded, more cross platform.	Owned by Microsoft. Mac strategy is unclear. Linux and embedded support is lacking.
Offering		Great for mobile only development, or to add mobile to Windows app	Qt is a much wider and more complete SW dev offering. True cross platform!	Proper cross platform is more expensive than Qt

Xamarin – Main Pitch

› Xamarin is only good for Mobile only applications, and has no support or strategy for embedded.

› For more than a single OS, Xamarin is more expensive than Qt!

Xamarin – main comparison points with Qt

Xamarin

- No support for embedded platforms, Linux.
  - Not focused on device Creation at all
- Uses native UI elements from the platform you are targeting
  - Can look more native on mobile
  - Needs different UI code for Android and iOS
  - Look and feel is not the same on different devices
- Very limited support for desktop
  - It is possible, but it is not meant for desktop.
- Owned by Microsoft. Uncertain future
  - Will it be cross platform
  - How will prices change?

Qt

- Extensive support and tools for building embedded devices
- Qt uses custom UI elements. Allows for pixel perfect UI creation.
  - Great support for Corporate Identity across devices
  - Much simpler UI creation. Faster time to market.
  - Pixel perfect UI from device to device
- Great support for desktop and embedded!
- Independent technology
  - Qt will listen to the users
  - Qt is growing fast and adding great features all the time

Crank

	Status	Stronger than Qt	Benefit of Qt	How to address this competitor
User Experience				
UI Design Experience		Fluid tooling	Versatile tooling	Very limited.
SW Developer Experience		Not at all! They barely have a SW framework.	Qt Creator, Amazing C++ API's, full featured framework	Crank is not really a SW framework, only a thin UI layer. Qt as an integrated solution enables shorter time to market for our customers.
Cross Platform		Limited, closed ecosystem	QPA, open source	Qt is more flexible and more futureproof.
Performance			Open, stable, high UI and application performance!	Runs in their engine. No way to introspect or fix issues yourself (as far as Nils can tell)
Footprint		Small footprint	Qt Lite if you want minimal	Narrow band of features. Only a thin UI layer.
Quality		Few references. Depends on back end + architecture	High quality. That is why it is used all over the place	Qt is an end to end solution, while Crank is the UI layer in a jig saw puzzle.
SDK		Nothing but UI tools	Qt Creator solves anything!	
Offering		Limited	Extensive, open, supported!	What about long term maintenance?
Ecosystem		Small, but connected to QNX and RTOS circles	Large, versatile, efficient. Get the job done, many devs!	Qt has a much larger ecosystem, and is a much more proven and complete solution!

Crank – main pitch points

- Crank is UI/UX only. Does not provide a framework to integrate all the development needs you have.
- They claim their one-size-fits-all solution is better than Qt, which is certainly not the case!
- Qt has multiple UI solutions to solve your problems in the best possible way!
  - **Qt Widgets** for a native and well integrated UI solution on desktop
  - **Qt Quick/QML** for an high performance, optimized, pixel-perfect UI on any device
  - **Qt Quick Components**, **Qt Charts**, **Virtual Keyboard**, **3D Data Visualization** for quick implementation of common concepts!
  - **Qt Canvas 3D** for simpler, webGL based applications. Ideal for IoT or Cloud based solutions.
  - **Qt 3D** for advanced and high performance 2,5D or 3D user interfaces
  - **Qt 3D Studio** to quickly and easily integrate the design process, and use common designer tools in the UI creation
  - Full support for native OpenGL, **Streaming OpenGL** and other innovative and modern UI approaches!

- There is a large ecosystem of Qt users, and a sustainable long term business model.
- Qt does not force one solution on our customers, but we provide all the parts needed!
- Qt offers professional support to make the right decision, and implement the optimal solution!

Android

Qt is behind	Same-isch	Qt is better	Qt is superior

	Status	Focus of Android	Benefit of Qt	How to address this competitor
User Experience				
UI Design Experience				Qt has a lot of tools and features to build amazing User Experiences. It is simpler to build a nice UI/UX with Qt.
SW Developer Experience		Well integrated tools, Java framework, and all you need, out of the box!	Qt is a better API, more efficient,	
Cross Platform		Not cross platform	Cross platform	
Performance		Qt applications have been shown to have better UI performance than Android Java apps.		
Footprint		N/A – but Qt will add a little bit to an Android system.		
Quality		Well maintained by Google. Two different parts, that are collaborative, more than competitive.		Most likely the HW vendor dictates the OS. Android has many benefits over maintaining your own Linux distro.
SDK		Their own SDK	Qt Creator	No conflict, but Qt adds more options, and is a better environment .
Offering				Qt allows Android based projects to be cross platform

Android – Overview

- Android is a version of Linux. Created by Google and made specifically for Mobile phones.
  - Android is not really a competitor to Qt. Qt relies on an Operating System (OS) and Android is a supported OS.
  - There are multiple versions tailored for other things, like IoT devices, and the automotive sector
  - Adds a lot of OS features and apps, and comes with Java as the developer offering.
  - Can also easily run Qt!

- Two main ways of using Android:
  - AOSP – the Android Open Source Project
    - No fees and no tie in with Google. You are on your own. No access to the Android Play store, or automatic updates
  - Google Mobile Services (GMS)
    - Google Chrome, GMail, YouTube, Play Store, etc. These are not open source!
    - Only available under a strict license from Google

Android – main pitch

› AOSP is a powerful way of building devices, and Qt provides the simplest, most cost efficient way of developing applications for such devices.

› Google Mobile Services is closed, expensive, and locks you in to Google's ecosystem. Qt is a simple way to do all the things you need, without vendor lock-in.

› Mobile apps can be written in Java or Qt. Qt is fully cross platform, but the Android Java API is not.

Kanzi

	Status	Stronger than Qt	Benefit of Qt	How to address this competitor
User Experience		One	Qt has more 2D controls	
UI Design Experience		One Designer for both 2D/3D.	No native desktop styling.	Qt has multiple UI approaches Not one size fits all
SW Developer Experience		Not a SW development tool at all. Also, a very limited ecosystem.	No IDE, just GUI designer Not an application tool, only the thin UI layer can be done.	Qt 3D studio adds to Qt everything they already have. Qt is a much stronger UI and application offering, solving all the development needs.
Cross Platform			No native APIs as it is in Qt Run-time based on C++98	
Performance		Proven solution.	Some controls are slow	Qt Quick rendering optimizations
Footprint		As small as 200 kB Kanzi lite is very small	Qt is larger, but has a lot more features. Qt Lite enables smaller builds than in the past	Not a complete framework like Qt is. Kanzi is only the UI piece of the puzzle! Qt is a full framework =>improves productivity!
Quality		This is a black box we do not know much about		Qt has many times larger community behind it
SDK		Installation experience	SDK is basically just a Designer	
Offering		Strong market position in Automotive clusters	Kanzi is only for automotive (embedded)	Qt Automotive Suite is adding all the building blocks to use Qt in the same areas. But Qt is more flexible. Qt knowledge can be used more.

Kanzi – main competitive points

Kanzi is a Automotive specific toolkit. This is a generic overview, but the Automotive team will provide more specific, in-depth information. Talk to Alistair Adams for details.

- **Kanzi Lite** is smaller than Qt, but is a thin UI layer, and does not provide all the necessary features to build your device, so their numbers are not representative for the whole device.
- We have heard that some of the Qt Quick controls are slower than the Kanzi control's
  - Not a viable argument. Qt can have better perofmance
- **Qt Lite** allows for a good balance between size and functionality
  - Small size
- Qt Quick has matured and can easily be as efficient as Kanzi today.

Kanzi – Main Pitch

- Kanzi is a very small ecosystem, with very limited knowledge and available experience. Finding developers is hard, and they are more expensive. This is a big business risk.
- UI Design tool only. The price of a Kanzi project is much higher, as you have to integrate with, and purchase other technologies as well to build a full solution.
- Automotive only is a single point of failure. Too few customers to be a resilient company.
- Kanzi has a much smaller market, and is a bigger risk of not providing support in a few years from now.
  - No support when they go out of business

Java

	Status	Stronger than Qt	Benefit of Qt	How to address this competitor
User Experience		More widely used	Qt is a better API.	
UI Design Experience			Qt has clearly better UI design tools and features	There are some add on's or extentions available, but in itself Java is not great for UI creation.
SW Developer Experience		Java can be faster to learn. Garbage collector for convenience.	Qt is a more productive API, more efficient applications and has predictable performance	Productivity in Java quickly hits the ceiling, and the cost of maintenance is too high.
Cross Platform		Wide support, but sand boxed environment.	Natively compiled, fast, safe, predictable,	Qt is far more efficient, and utilizes the underlying platform much better
Performance		Java makes it simple to write apps, but hard to optimize performance	No garbage collector. C++ beats Java to a pulp every time! Qt is faster and more predictable.	The code model, and parent child relationship in Qt makes memory management simple. Performance is simple to make better in Qt!
Footprint		High. Uses dynamically shared libraries. Few options to optimize	Shared or dynamically linked. Qt Lite allows for optimizations.	Qt foot print can be minimized and optimized for varioys use cases. No need to deploy a full virtual machine.
Quality			Simple to write high quality code	
SDK		Many IDE's and tools available	Qt Creator is amazing, but any editor can be used. Better API	
Offering		Free, but be aware of Orcale		Many licensing issues with Java, and Google is in the shit.

Java – What is the difference from Qt

Java

- Commonly used and thought in universities.
  - Object orientation and easy to learn
  - Widely used for application development
- The Java API's are not well designed
  - Large projects become extremely hard to maintain
  - Development is slow and inefficient
- Code is complex and too "verbose"
  - Managers like this, as it looks like "a lot gets done"
  - Projects very quickly grow in size.
  - Maintenance becomes expensive and difficult

Qt– counter arguments

- Qt is open source, widely used, easy to learn
  - Qt takes more time to learn, but is a more productive environment
- High quality API's
  - Developers write code faster, debug code more easily
  - Maintenance of projects cost less
  - Easier to switch developers on the project
- Qt code is efficient and much more readable
- Qt Projects have lower total cost of ownership

Java – What is the difference from Qt

Java

- Java is generally disliked by many customers
  - Bad reputations due to invasive upgrades
- Java look and feel is not great on desktop
- Java is a sandboxed in a virtual machine
  - Impossible to get the same optimization as with Qt
  - Hardware access, peripherals and drivers is difficult
- “Managed code is easier to write”
  - Java has a garbage collector
  - Makes the code is slow and unpredictable

Qt – counter arguments

- Qt has no negative consumer connotations
  - Small independent Qt is more liked than Oracle
- Qt is either native of custom
  - Well liked Look and Feel
- Completely native framework is more flexible
  - Simple to access all features of the machine
  - Easy to get access to all HW extensions, drivers, etc.
- Qt and C++ is not managed
  - Qt provides a design that does not need a garbage collector, but is still very simple to use. Parent/Child model is very powerful and simple to use.

Eclipse vs. Qt Creator

Eclipse – Java based IDE

- Large ecosystem
- No clear direction, but something for everyone
- No company investing significantly
- Pulled in all directions, by Oracle and many others.
- Many forks for various purposes

Qt Creator

- Created specifically for Qt development
  - Very good at this!
- Open Source, and open plugin architecture
  - Use and modify Qt Creator as you want to!
  - Very flexible and powerful

Java – Main Pitches

- Java is an inefficient language.
  - Developers need to write too much code, and the cost of maintenance is much higher than with Qt.
  - Time to market is slower than with Qt
- Java is loosing traction, having fewer users and less impact.
- Java is not a good embedded technology! If embedded is a potential target, Qt is much better!

Embarcadero Delphi

	Status	Stronger than Qt	Weaker than Qt	How to address this competitor
User Experience				
UI Design Experience		Multiple UI libraries, and market place of UI components[1]	Imperative and declarative UI paradigms. Simple to create custom User Interfaces	
SW Developer Experience			Embarcadero has many languages and features, but this quickly gets messy	No development possible on Mac. Only a deployment platform.
Cross Platform		Multiple features that are native cross platform	Only some tools can target each platform. Strange mix	Qt is a more unified product, with all features fully cross platform!
Performance				
Footprint				No info at this point
Quality				
SDK		RAD studio seems OK	Qt Creator is perfect for Qt	
Offering		Many components and options	Messy and complicated offering, but targets many use cases and has a lot of features.	Specialized language, small user ecosystem, no open source ecosystem.

Embarcadero Delphi

- Cross platform development framework
  - UI: VCL component library (for Windows) or the smart cross-platform FireMonkey (FMX) library
  - Ecosystem of commercial and open source components, available through the IDE GetIt package manager
- It is a lot harder to develop a custom UI than it is with Qt

- Small or non-existent community. No open source version. No external language bindings

- Developed by various companies, for different goals over the years. Supports many things, but seems to lack a grand unifying design philosophy or goal.
- Business focused, more than developer focused.

Delphi – Main Pitches

- Closed source framework – no transparency, no ecosystem
  - No way of easily judging the quality of the framework
- A collection of business tools, developed to make money, not developed out out SW developers needs
-

GTK+

	Status	Stronger than Qt	Benefit of Qt	How to address this competitor
User Experience				
UI Design Experience		Widget only toolkit	Qt Creator and Qt Quick Designer supports a much wider set of UI technologies	
SW Developer Experience				
Cross Platform		No, but has about the same capabilities as Qt	Wider support for embedded platforms and RTOS.	
Performance				
Footprint			GTK+ has been criticized for being very memory hungry.	
Quality			They have been breaking Binary compatibility in many of their releases.	Not developed by a company, but a pure open source project. No centralized management, and very little regard for commercial priorities.
SDK				
Offering		Free of charge	Much better business model	Qt is a more reliable framework, developed by a company, with business applications in mind!

GTK+

- Cross platform, widget based UI and application development framework.
- GTK is best for old school, static, desktop centric User Interfaces – Qt is better for anything more future proof.

- Limited support and functionality outside the Linux world.

- Free software available under LGPL 2.1
  - No business model. Fewer developers. Moving slowly.

- Open source project riddled with conflicts and controversy, no clear leadership and direction
  - Qt Project has Lars Knoll (Qt Company CTO) at the top, making all decissions. Prioritizes Qt customers!

Enlightenment Foundation Toolkit (EFL)

	Status	Stronger than Qt	Benefit of Qt	How to address this competitor
User Experience				
UI Design Experience				Designing a simple UI in Qt takes only hours, in EFL it takes months.
SW Developer Experience		C based. Library names don't mean anything,	Several EFL developers have written at length about how bad this framework is.	Bring up the writings and testimonials of developers who had to use EFL for projects.
Cross Platform		EFL is not cross platform, it only runs on Linux.		
Performance				
Footprint				
Quality		Qt has a much more well designed API, and is easier to use in a correct way!		Qt is a better framework for commercial and business critical applications.
SDK		They have multiple IDE's, but none seem to have good quality		
Offering			Proper business model, and a solid,	

EFL – The Enlightenment Foundation Toolkit

- EFL is a C based project, that tries to do many of the same things that Qt does.
- EFL is based on C, but adds some sort of Object Orientation on top of it
  - This is not a good architecture, and leads to many challenges.
  - Qt is C++ and has a solid and efficient architecture. More stable, larger ecosystem and a better programing paradigm.
- No commercial revenue model – future of development is not certain!
  - Funded in part by Samsung, and a few other benefactors, mainly to be used on Tizen
  - Most of the products in the market are Samsung products, very limited usage.
  - Small ecosystem. Very few developers of the framework, and very few users
  - If Samsung drops Tizen, the funding of EFL is very uncertain. Not a stable product.

- Quote from an EFL developer
  - "Some of you may know a Qt presentation, where someone built a fully functional media player with theme support in QML in real-time in front of an audience. Doing the same in EFL is measured in man-*years*. I'm pretty serious about that one. Small projects like these require teams of 5-10 programmers working up to a year."

EFL – Main Pitch

› EFL is a small ecosystem, limited to Samsung only.

› The technology is so poort, that even despite heavy investment and free access, no one has been interested in starting to use it!

› The developers using it are loudly complaining about quality in public forums.

› There is no cross platform support.

› The future maintanance and development of EFL is uncertain, because there is no commercial revenue model.

vxWidgets

	Status	Stronger than Qt	Benefit of Qt	How to address this competitor
User Experience				
UI Design Experience		This is a traditional widget based framework.	Nothing that compares to QML or other modern UI ideas	Qt has a lot more features, tools, and produces much more professional and modern UI's
SW Developer Experience		Python or C++, lots of documentation and demos		Depends on GTK+ on Linux, which is not a good architecture.
Cross Platform		Cross platform, UNIX focus	Wider platform support (RTOS)	Qt has wider, more updated x-platform support
Performance	Same			
Footprint	Same			
Quality				Qt is better. With more testing, larger ecosystem, more releases, LTS-support etc.
SDK		No dedicated IDE.	Qt Creator with all the integrated tools and features.	Qt has a much more complete SDK, with Qt Creator, pre compiled binaries, pre compiled embedded targets, etc.
Offering		Open Source, LGPL with static linking permission. Support through 3rd party	vxWidgets is a similar, but less comprehensive and less mature version of Qt.	Qt is a much more complete and mature offering, faster product development and better customer service.

vxWidgets vs. Qt

- Only about 20 people are contributing actively to the vxWidgets code.
- Very old technology, that has not been kept up to date, or kept relevant.
- Qt is a large ecosystem with about 150full time contributors.
- Qt is also true and tested, but Qt is in active development and akways adding modern features like QML and Web Engine support

vxWidgets

› Small ecosystem that is slowly crawling along.

› Few new features

› Security and stability is less secure, as this is much less used than Qt.

Altia

	Status	Stronger than Qt	Benefit of Qt	How to address this competitor
User Experience				
UI Design Experience				
SW Developer Experience				
Cross Platform				
Performance				
Footprint				
Quality				
SDK				
Offering				

Altia - overview

- Altia looks to me to be a simple to use, HMI solution.
- Thin UI layer only, and not much more. (-)
- Their GUI designer generates C code. (+)
- Good Cross platform support; RTOS, or no OS at all, because it is only graphics, and based on C (+)
- Customer support available in multiple languages: English, German, Italian, Japanese and Mandarin. (+)
-

Altia

Altia is a Automotive and Medical specific toolkit. This is a generic overview, but the Medical team will provide more specific, in-depth information. Talk to Roger Mazala or Alistair Adams for details.

- Altia is stronger because:
- Qt end to end UI design and tooling suite is not very strong.
  - Qt does not have Photoshop, Adobe or other design tools integration
  - Altia's PhotoProto integrates with PhotoShop
- Footprint of Altia is much smaller
  - But we are not sure what the footprint of a complete solution would be
- Altia focuses on tooling and code generation
  - validated code as the output of these tools
- Altia has a wider set of support HW, out of the box, in the embedded space.

- Qt is stronger because:
- We have a much more complete solution, with a wider set of functionalities.
- More efficient ecosystem, better integration between UI and functionality
- You can easily get Qt running on a device we do not "support".

Altia – Main Pitches

› UI Design tool only. The price of an Altia project is much higher than they pretend, as you have to integrate with, and purchase other technologies as well to build a full solution.

Microsoft Tools

	Status	Stronger than Qt in	Weaker than Qt in	Qt messaging
User Experience				
UI Design Experience				
SW Developer Experience				
Cross Platform				
Performance				
Footprint				
Quality				
SDK				
Offering				

Microsoft tools overview

- Microsoft has no history with open source or cross platform.
  - Minor parts are getting open sourced, but most critical components are still closed
- Visual Studio is a bundle of C++, .Net, MSVC compiler, IDE and many other things
  - Fixed set of tools and workflow that is not a good fit
- Incompatibilities between MSCV versions
  - Using more than one version of Windows, or one version of any tool, is a large challenge with MSVC
- .Net & Visual C++ depends on commercial tools
- Contains and actively uses SW patents
- UI is specific for Windows

- Qt has always been fully open source, and cross platform.
  - Full transparency, full flexibility
- Develop amazing applications using Visual Studio, Qt Creator or any other tool
  - Happy developers are productive developers
- Qt provides great cross-Windows support,
  - Many Windows-only project are using Qt for it's unparalleled binary compatibility and support!
- Qt can use, but does not depend on MSVC
- Free of patented SW, uses only SW licensing
- Pixel perfect UI across any Operating System
  - Qt Quick is a fast and simple way to develop amazing UI's for any device, screen size or operating system.

Microsoft tools overview

- Subscription license only
  - Long term usage is more expensive than it might look
- Microsoft is a large and heavy organization
  - Getting visibility is expensive, and getting bugs fixes can be time consuming
  - Not possible to fix anything yourself, as it is mainly closed source

- Qt has a perpetual license
  - Subscription also available if needed
- Every customer is valued and important
  - Support team work directly with R&D
  - We fixe and integrate changes that are important to you!

- Qt has best in class documentation, and is faster to learn than other C++ frameworks.
- Easily produces clean code that is fast to learn, and simple to validate or certify.

Microsoft Visual C++ (MSVC)

- Microsoft Visual C++ has not been a stable ABI
- Not an Open Source implementation of C++
- Minor parts are getting open sourced, but most critical components are still closed source
- Visual C++ 2017 support C++ 11

- Qt has always been fully open source, and cross platform.
  - Full transparency, full flexibility
- Develop amazing applications using Visual Studio, Qt Creator or any other tool
  - Happy developers are productive developers
- Qt provides great cross-Windows support,
  - Many Windows-only project are using Qt for it's unparalleled binary compatibility and support!
- Qt can use, but does not depend on MSVC
- Free of patented SW, uses only SW licensing
- Pixel perfect UI across any Operating System
  - Qt Quick is a fast and simple way to develop amazing UI's for any device, screen size or operating system.

Stand-alone C++

	Status	Focus of C++	Focus of Qt	Qt messaging
User Experience		A little bit hard to learn	Easy to learn, easy to master!	
UI Design Experience		This is not the focus or capability of C++	Qt Quick, QML, Qt Widgets, HTML, you can do it all!	Qt *is* the way to do UI's in C++
SW Developer Experience		Low level, powerful but hard to use .	Powerful, but Qt adds ease of use, UI and documentation	
Cross Platform		Yes	Yes	
Performance		Yes	Yes, but it is simpler to achieve it with Qt	
Footprint		Smaller than Qt	Adds functionality that is needed and wanted	
Quality		Best in the world	Best in the world!	
SDK		No good SDK around it. Many choices.	Qt is really the best C++ based SDK.	
Offering		Only a language	A complete, well supported offering	

Stand Alone C++

- Qt provides a much nicer and easier API to work with
- The Parent / Child model in Qt makes efficient development a lot faster and reduces the amount of memory errors.
- The Qt Object model makes debugging, introspection and understanding your code much more efficient.

Other technologies and frameworks to consider

Desktop and mobile

- Electron
- PCB software

- Unity
- Swift
- Adobe Creative Cloud Suite

- Deksi
- Power Bio

Embedded

- Embedded Wizard
  - http://www.embedded-wizard.de/
  - Nice looking demo video! Not nearly as powerful as Qt
- TouchGFX
- IAR (not really a competitor)
  - They use Eclipse for their IDE – NC has tried to move them to Qt
  - Embedded up to Cortex M4/7 as the very high end

Web Technologies

Introduction

- The content of the next slides is to provide a factual and realistic description of the strengths and benefits of HTML5 vs. Qt
- This is mainly meant to enable you to ask the right questions from the customer, and make them realize the limitations of HTML
- The benefits of Qt are clear when contrasted with this view of HTML5 and browser based solutions

Web Frameworks: Overview

- Common things that evaluators of Qt can say are benefits, include
  - Many developers know HTML5 and the corresponding technologies
  - It is “cross platform”, and will run anywhere
  - We can build a SW development platform on top of it, and get third party apps

- Both we, and usually also they, know that the main arguments against any kind of web based framework are usually most or all of the following
  - User Interface performance and overall performance of the solution
  - Native access to HW and drivers, input mechanisms or more than one screen
  - Maintainability and longer term evolution of the application
  - Memory usage and overall application footprint
  - Look and Feel
  - Scalability
  - Predictable behavior

Web Frameworks: Developer Ecosystem

- The customer will never get a wide range of high quality apps for “free” from the ecosystem.
  - As HTML5 is free to use, and there are many developers who know how to use it, some companies think hundreds of apps magically we be created by the community for their platform. This is almost never the case.
  - The first few thousand iPhone apps were mainly low quality, advertisement generators of low value and quality. That is mainly what the greater ecosystem provides for free.

- You need to define, pay for (one way or another) and carefully maintain high quality apps
  - If you commission a Spotify, Netflix, or other custom app for your embedded platform, skilled developers can use almost any framework. But not all technologies can deliver highly performing apps.
  - Focus on a few, trusted, high-value, high-performance apps, that the customers need.

- Use Case example: LG is using Qt for their smart TV’s.
  - They provide HTML as a developer offering for the ecosystem, but you can immediately see which apps are HTML based. HTML5 apps are slower, and not as nice looking as the native apps that are all done in QML. (Source: I own a 4K LG TV)

Web Frameworks: Performance

- User interfaces implemented mainly or fully in JavaScript and HTML5 are heavy, and needs powerful and expensive HW to look OK.
- Facebook failed to create an iPhone-like user experience on smartphones that are much more powerful than the system-on-chip (SoC) used in most premium embedded devices appliances.
- Despite an enormous engineering effort, Netflix was not able to create an iPhone-like user experience on TVs and Set Top Boxes with much more powerful hardware than most cost conscious devices.
- These are large companies, with many highly skilled developers, and they threw away a lot of money, and both backed down on the HTML5 strategy, and implemented their applications using native frameworks.

- Is the customer's organization able to do better than these guys? Or do they have more money to spend than these organizations?

Web Frameworks: Cross Platform

- This is a common misconception, especially for embedded
- You need a web engine to power your HTML based application.
  - There are few options for embedded devices
  - Most commercial embedded browser are more expensive solutions than Qt (including dev seats + distribution licensing)
- Integrating HW specific capabilities with your web engine is hard, and costly.
- Web Engines and browsers are complex, and expensive to develop.
  - Fixing bugs is harder and more expensive than for a native application, as the complexity is higher

Web Frameworks: Support and Maintenance

- Long term support of HTML
  - HTML is a fast moving technology, and the features you rely on may quickly become unsupported
  - The long term support cost can be significantly higher
- Browser implementations are different for OS and HW platforms
  - You need to provide support and maintenance for multiple different implementations
  - This drives the cost of maintenance and support up
- Maintenance of large applications becomes a nightmare over time
  - HTML lack fundamental features to support large SW projects. Like object orientation, global variables, a thread model, memory management and code separation.

- Long term support of HTML
  - Qt is a stable and solid technology
  - Qt provide LTS releases
  - Qt provides special, commercial, support as need be
- Qt is cross platform
  - Qt can run on a wide number of hardware and operating system platforms
  - UI and feature implementation is done in C++
- Designed for large SW projects
  - Clean model for business logic and UI separation
  - Process model for high performance and a smooth UI
  - Object oriented highly efficient C++ code
  - Fully transparent memory management features
  - Components

Web Frameworks: Cost comparison

- To achieve the same performance, look and feel, hardware cost and overall customer satisfaction, the total cost of ownership is much higher with HTML5 compared to Qt Commercial.
- To drive a nice UI, you need a more powerful HW. Even for low distribution volumes, this quickly offsets the license pricing of Qt
  - A theoretical customer have to use an SOC with higher performance to utilize HTML at the same performance.
  - The price difference between an i.MX53 and an i.MX6 (one league higher) is more than 11 Euros when bought in batches of 1 million. This does not even consider the additional costs for more RAM and flash memory needed by HTML5 apps – compared to Qt apps.
  - If we are more conservative, we can reduce the price difference to around 3 euros / dollars for a HW that can power HTML5
  - For a high volume of devices, this is quickly a price difference of multiple million dollars.
- Both Netflix and Facebook made multi-million dollar mistakes with HTML5!

Testimonials

› Qt is far more portable and simple to support across hardware architectures and devices, and over a long period of time than HTML is. HTML is moving too fast, and the difference in browser and web engine implementation is too big.
– Qt Consultant (but it is 100% true)

› The biggest mistake that we made as a company is betting too much on HTML5 as opposed to native [...] We burned two years.”
– Mark Zuckerberg

Qt vs. HTML 5 – Main Pitch

- HTML5 is more expensive
  - the devices needed to power the same UX cost a lot more
  - Maintenance is more expensive
  - Security is more difficult and expensive – The web engine changes rapidly and is a voulnerable piece of SW
- It is easier and cheaper to build a great User Experience with Qt

- The HTML5 ecosystem is larger, but you will never get high quality apps for free. You need skilled developers to create your content.

HTML 5 in general

› According to Siemens, Qt is up to 14 times more efficient than HTML5