Intro to Qt

Kevin Krammer

February 18

There has never been a more bountiful time for developers than there is right now. There are more languages, frameworks, and libraries available than ever before, with many of them free or nearly so. But every new tool comes at a cost.

We're talking about learning curves and initial configurations as well as the tasks of finding experts or mentors, and sourcing compatible components. And then, once you get off the ground – then what? What if your tool can't stay relevant? A code base can create a huge amount of inertia that will keep you in that tool's orbit, even when it clearly becomes obsolete.

So despite the number of choices, it's still critical to make the right choice. At KDAB, we think that choice is Qt. We'll discuss some of the reasons we think Qt is a worthwhile time investment and the circumstances in which Qt shines so you can decide for yourself if it's right for you. We think you'll agree with us that Qt is a great choice so we've also got a few simple steps on how to get started and where you can go for additional resources.

Picking up a new tool

If you're starting a new project, it may be the right time to choose a new tool but the field is crowded with options fighting for developer mindshare. Here's what we think are the most common things you'll want to consider:

- Ability to target as many platforms as possible – Windows PCs, Mac OS, Linux, iOS and Android, phones and tablets

Qt is designed to provide a consistent GUI on all platforms, running on all desktop, mobile and nearly all embedded hardware.

- Support for embedded devices like i.Mx6, Arduino, Raspberry Pi, Beaglebone, or Intel Edson

- Ability to create an attractive modern user interface

Seems simple enough, right? Let's take a very quick survey of the most popular cross-platform candidates and see how well they meet these criteria.

Java: This well-developed language has lots of support for the Android platform and desktops, but getting the JVM running on iOS or embedded boards isn't a ready-made easy experience. Android has a standardized UI framework; all other platforms can choose among a number of options with various support/speed/complexity trade-offs. Intermittent garbage collection may create pauses or stutters in the user interface. On embedded platforms, accessing the hardware requires JNI callbacks due to the JVM's design.

HTML5: This is an exclusive tool for web development and so there are many HTML5 frameworks available. Most are built for web support only, although some (like Adobe PhoneGap) target mobile development. HTML5 is inherently cross-platform and visually oriented, and provides excellent support for styling and theming. However browser incompatibilities (Chrome, Safari, IE, Firefox, WebKit, etc) may make the app behave or appear slightly differently on different platforms, and per-platform browser testing is required. Interpreted JavaScript and complex interactions of HTML, CSS, and JavaScript result in large memory footprints and may provide sluggish performance. By-design, it's unable to access the underlying hardware and middleware functionality for embedded applications.

C++: This is a powerful and mature language able to target any modern desktop, mobile, or embedded platform. C++ provides highly optimized compilers with best runtime performance and tightest memory control of all commonly used languages. Modern additions to the C++ language with C++11/14/17 have improved type safety, capability, and ease of use. It's easy to access underlying hardware on embedded systems. It's stable and scalable for large projects. It does not have a UI option built-in, so one must be added.

Complex animations are trivial to do with QML

Qt: This comprehensive open-community framework is built on top of C++ and thus inherits the benefits of C++ listed above but also adds run-time types, dynamic properties, and GUI-event handling. Designed to provide a consistent GUI on all platforms, it runs on all desktop, mobile and nearly all embedded hardware. It offers simple declarative user interface creation with QML. It comes complete with many varieties of graphical objects, including traditional desktop controls (Qt Widgets), 3D graphics, and data visualization charts/graphs. With it, developers can create fluid animations on constrained hardware.

Qt's capability for developing high performance and scalable software means with one tool you can develop anything from simple desktop apps to large enterprise systems.

Qt is used for all types of aaplications including in-flight systems

Choosing the best option

At KDAB, we consider Qt to be the overall best option for most application development, particularly whenever there is a UI requirement. There are a number of reasons for this.

- **Multi-platform**: Developers can target multiple platforms with the same code base
- **Performance**: Compact, high-performance applications are the norm
- **Innovation**: Qt focuses on innovation rather than infrastructure coding
- **Licensing**: Flexible licensing options abound: Commercial, LGPL, and GPL
- **Professional services, support, and training**: There's plenty of all three
- **Breadth and depth**: Developers can find nearly anything they need

Almost no matter what kind of development you're thinking about, you'll be able to do it with Qt. Being the strongest cross-platform tool, Qt makes your applications available to the largest number of potential users for desktops, mobiles, and embedded devices. And because of Qt's capability for developing high performance and scalable software, with one tool you can develop anything from simple desktop apps to large enterprise systems.

Addressing ease of use

Some say C++ has a reputation for being hard to use but if so, wouldn't that mean building Qt apps is difficult? It's certainly fair to say that C++ is a tool for professionals and it's got a long history with lots of different ways to approach a problem. The newest standards C++11, C++14, and C++17 add functionality and simplify syntax to give C++ improvements based on years of developer feedback and many new features from other modern languages. And with Qt, developers can start with QtQuick and QML, which provide most of what's needed to build a modern UI with the simplicity of JavaScript, taking mere minutes to become productive. The Qt Hello World section below gives you an idea of how simple it is to get started.

Who else is using Qt?

Lots of companies are using Qt for all kinds of applications: Amazon Lumberyard, Google Earth, Spotify, and Wolfram Mathematica are just a few examples. You might see some movies over the weekend that have been helped by Qt since Walt Disney and Lucasfilm both use it. It's also been used in way too many embedded systems to count in all types of specialized niches, although LG's smart TVs, Panasonic's inflight systems, and Harman's car infotainment systems are among those that you might encounter on a regular basis. If you pick Qt, you're in good company.

Qt is a mature development environment with a number of IDE plug-ins and utilities that expand programmer capability and reduce code drudgery.

What's in Qt?

A Qt installation has everything you need to get started building apps right away, plus lots of components you might need in the future:

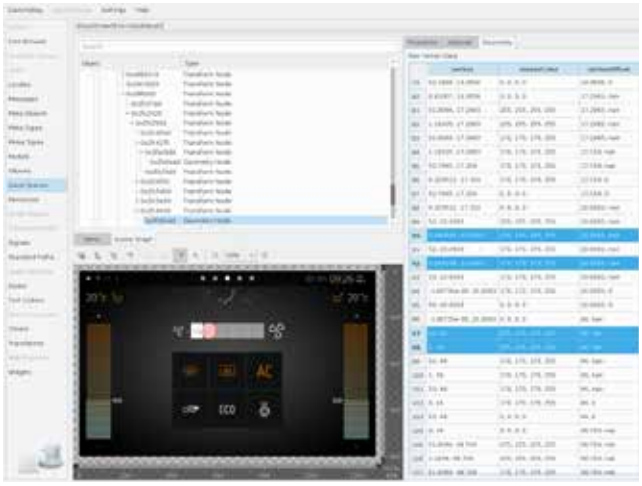
- **Qt Creator, qmake** – powerful Qt-aware IDE, build system
- **Qt Quick** – QML-based user interface design
- **Qt Widgets** – user interface design based on desktop controls
- **Qt WebView, Qt Web Engin**e – web containers
- **Qt SQL** – database integration layer
- **Qt Network** – network programming
- **Qt Multimedia** – audio, video, radio, and camera
- **Qt Purchasing, Qt Positioning, Qt NFC, Qt Sensors, Qt Bluetooth** – mobile platform-specific features
- **QtCharts, QtDataVisualization** – charting and visualization
- **QtVirtualKeyboard** – on-screen keyboard for touch screens
- **Qt Linguist** – internationalization and translation support
- **Qt 3D** – building 3D OpenGL apps

This list is just a sample and the available options are always growing. Visit this online resource for the latest: https://doc.qt.io/qt-5/qtmodules.html.

Qt Tools

As a mature development environment, Qt has a number IDE plug-ins and utilities that expand the capability of the programmer and reduce the drudgery in writing and debugging code:

- **GammaRay** – powerful debugging tool that lets you examine and change Qt objects at runtime
- **Clazy** – static analysis tool to identify where your Qt code could benefit from best practices
- **valgrind** – code correctness tool that finds things like memory leaks, uninitialized data, and thread race conditions
- **Hotspot** – graphical front-end for Linux execution/performance profiler
- And more



GammaRay can be used to inspect running programs

Qt modules, add-ons, and tools meet nearly every developer need for today's modern applications.

Getting your first Qt app up and running is so simple that it barely needs instruction; dozens of included tutorials make learning it a breeze.

Qt Hello World made dead simple

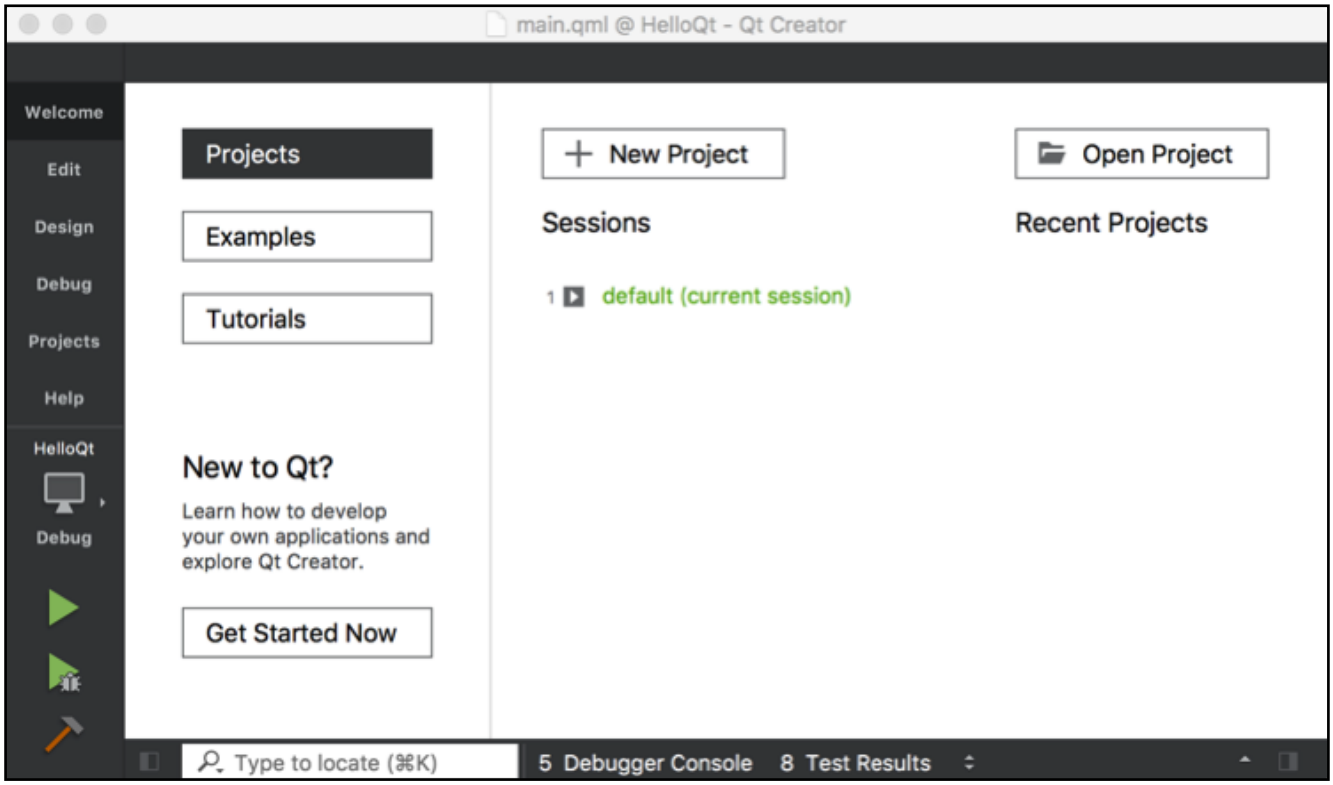
We've written down practically every click and keystroke needed to create a simple "Hello Qt" application, so you can see just how straightforward it is.

1. **Download and run Qt**

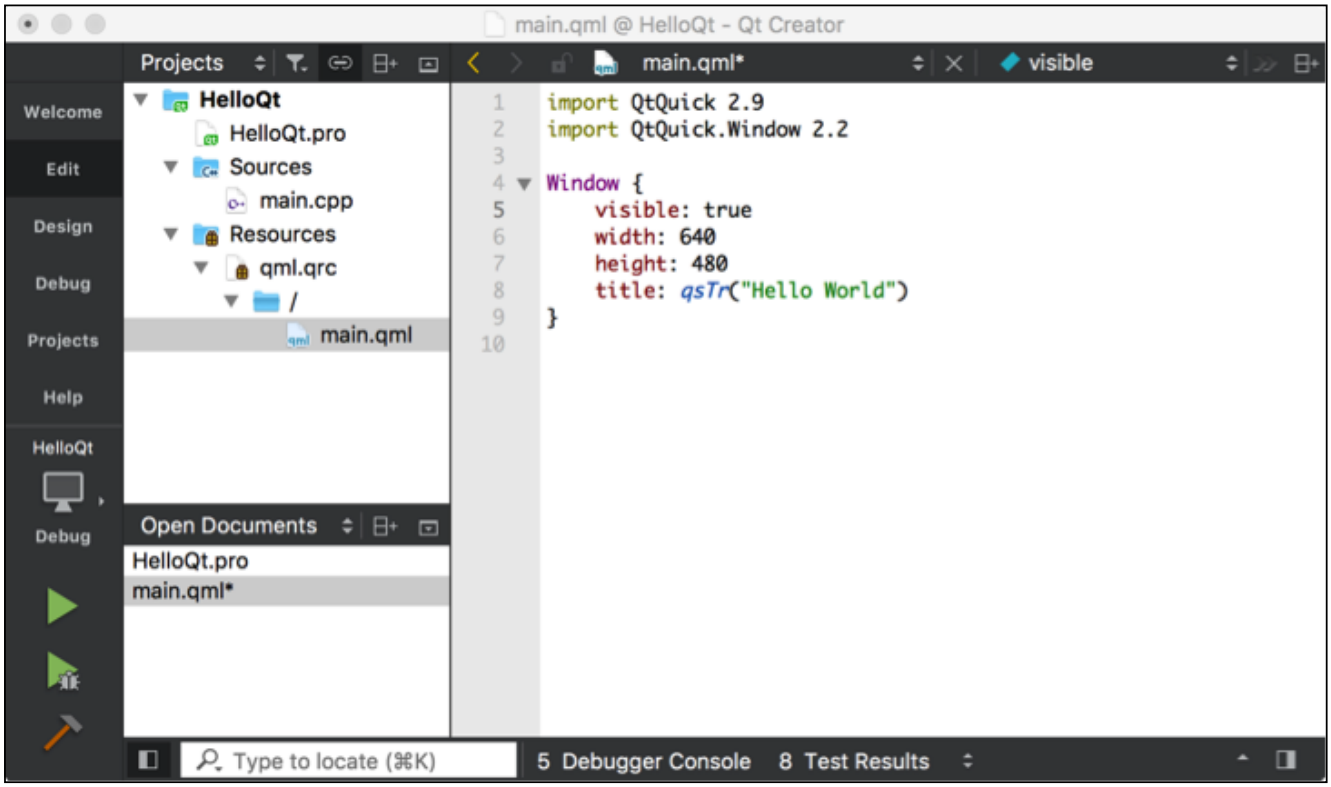
- **Download the Qt installer** from https://www.qt.io/download.
- **Enter an installation folder** or accept the default location.
- **Pick install options**, although the default installation choices should cover everything you need. Click "Continue" to start the install.
- **Installation** takes anywhere from 15 to 45 minutes depending on your computer, so it may be a good thing to get started before lunch.
- **Click "Launch Qt Creator"** once the install finishes to start up the IDE.

2. **Create a Qt Hello World**

- **Start a new project** by clicking the "+ New Project" button.
- **Pick the template** as "Qt Quick Application – Empty", and click "Choose".
- **Name your project** "HelloQt", leaving the rest of the fields with default values, and click "Continue".
- **Leave the defaults** for the next few screens (build system, version, kit selection), and click "Continue" on each screen.
- **Skip version control settings** and click "Done".



UX development in Qt can be completely done with graphical design tools, simple scripting, C++ code, or a mix of all three depending on your needs.



3. Add some code

- At this point, you've got a fully functional Qt project with an empty main window. The main user interface QML should look like the above code.

Developers typically enjoy the ease of programming that comes with using Qt

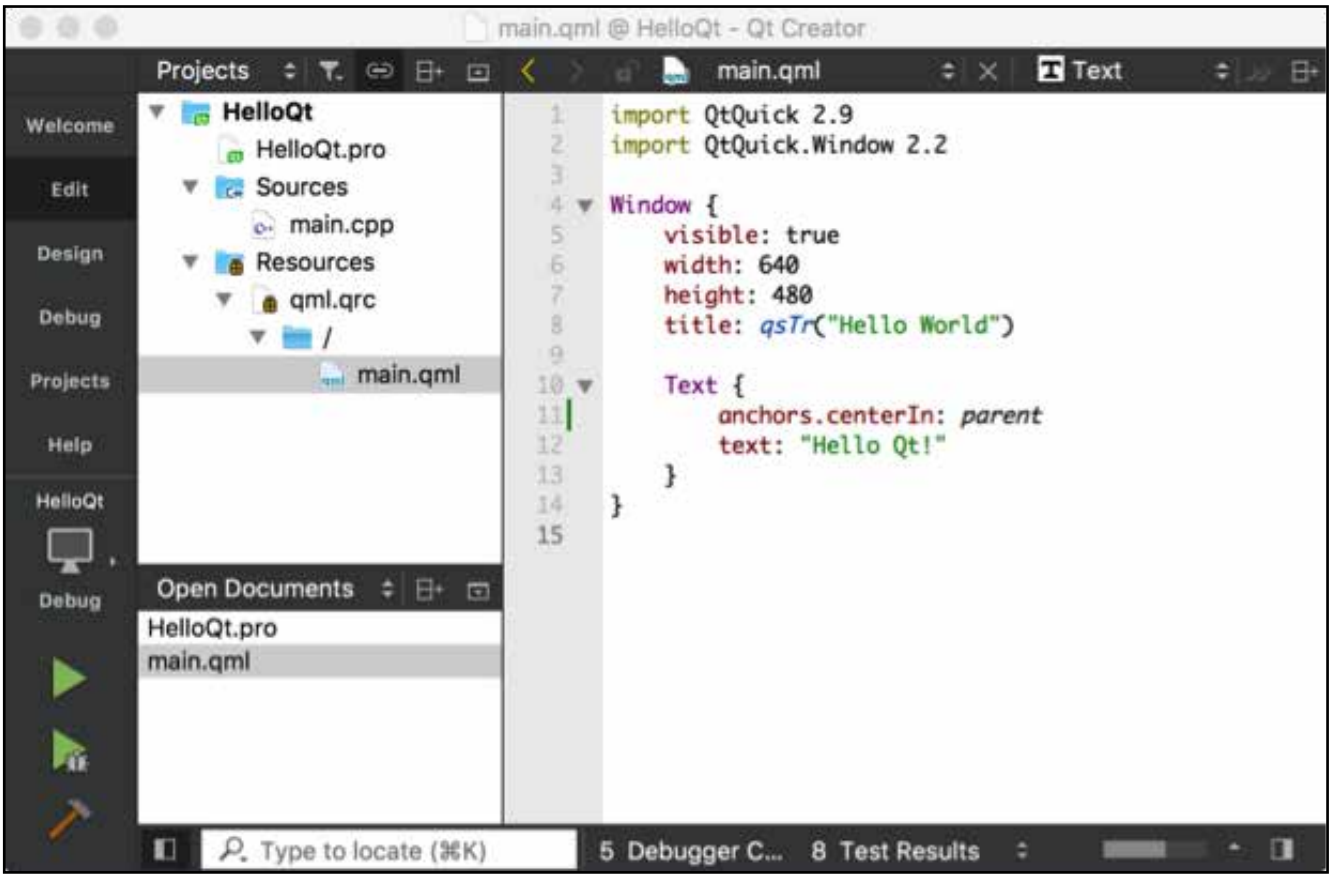
For a bit of excitement, let's add a "Hello Qt" label before we run it.

- **Add a text label.** You can do so in two ways. For a drag-and-drop method, click the Design tab, and drag a Text control onto the main.qml canvas, and type in "Hello Qt" into the Text field.

However, we're going to do it the "harder" way, by adding it in code to show actually how easy this is too. Click in the editor window and insert the following text between title and the last closing bracket:

```
Text {
   anchors.centerIn: parent
   text: "Hello Qt!"
}
```

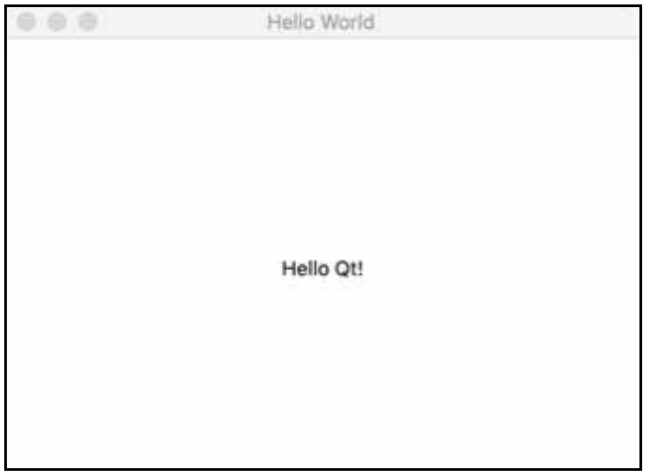
Qt has several licenses available that make it possible to create open source projects or proprietary customer software, and anything in-between.



Your code should now look like the above. Once it does, click **Ctrl+S** to save it.

- **Run it** by clicking the big green "Play" icon in the left toolbar. Done!

If you were successful, it should look like this:



From here on in, you've got a working code sample – you can try building it for mobile platforms, adding more snazzy graphics, or running one of the example projects. See the Welcome page Examples button for pages upon pages of interesting, colorful, and explanatory examples that cover many aspects of Qt.

Licensing

Most developers don't pay attention to licensing requirements until they need to ship their product. Unfortunately, that's when it can become a deal breaker. We don't need to get into the details here but thankfully Qt has several licenses available that make it possible to create open source projects or proprietary customer software, and anything in-between. Here's a guide for any licensing questions: http://qt.io/FAQ

There's plenty of assisstance that comes in a Qt installation, including documentation, tutorials, and hundreds of working examples.

Where to go for more

A quick look on Google will turn up plenty of resources but here are some of our favorites:

- Qt developer network: http://qt.io/
- Qt centre forum: http://www.qtcentre.org/
- Qt mailing lists: http://lists.qt-project.org/
- Training schedule: https://www.kdab.com/schedule/

Some great Qt tools to get familiar with:

- GammaRay introduction: https://www.kdab.com/what-is-gammaray/
- GammaRay : http://www.kdab.com/gammaray/
- Clazy : http://www.kdab.com/clazy/
- Hotspot: http://www.kdab.com/hotspot-gui-linux-perf-profiler/

Plus, don't forget the assistance that comes in a Qt installation:

- Documentation – getting started, using the IDE, running examples, API reference, and lots more (Qt Creator menu, Help | Index)
- Qt Tutorials – many hands-on videos for different topics (Qt Creator Welcome page)
- Examples – hundreds of working example projects (Qt Creator Welcome page, install dir $QTDIR/examples)

About the KDAB Group

The KDAB Group is the world's leading software consultancy for architecture, development and design of Qt, C++ and OpenGL applications across desktop, embedded and mobile platforms. KDAB is the biggest independent contributor to Qt and is the world's first ISO 9001 certified Qt consulting and development company. Our experts build run-times, mix native and web technologies, solve hardware stack performance issues and porting problems for hundreds of customers, many among the Fortune 500. KDAB's tools and extensive experience in creating, debugging, profiling and porting complex applications help developers worldwide to deliver successful projects. KDAB's trainers, all full-time developers, provide market leading, hands-on, training for Qt, OpenGL and modern C++ in multiple languages.

KDAB

CERTIFIED ISO 9001:2015 COMPANY

www.kdab.com