

A background image of a glowing lightbulb lying on its side against a dark, textured surface. The lightbulb is illuminated from within, creating a warm, golden glow that fades into the dark background. The base of the bulb is visible on the right side.

Qt For Beginners Training

Part 1

© Integrated Computer Solutions, Inc.
All Rights Reserved

Where to Download Qt?

- Commercial and open source licenses available
- <https://www.qt.io/download> (Online Installer)
- https://download.qt.io/official_releases/qt/
 - (Source and instructions on how to build)
- Many Linux distributions provide Qt as a native package
- Be aware of license requirements

What is Qt?

- Cross-Platform Software Toolkit and API that allows to develop World-Class Desktop and Embedded Devices applications
- Spelled Qt and not QT
- Pronounced “cute”
- Is owned by **The Qt Company**
- **The Qt Company** is responsible for all Qt activities including product development
- <https://resources.qt.io/customer-stories-all>

All Rights Reserved

Companies Using Qt



Panasonic

sky

SENNHEISER

© Integrated Computer Solutions, Inc.

All Rights Reserved



navico

RIMAC



Mercedes-Benz

ubuntu®

formlabs 

ABB

 Pitney Bowes

Where to Get Help?

- Documentation in Qt Assistant or Qt Creator
- Qt's examples: \$QTSRC/Examples
- Qt Project: <http://www.qt.io/developers/>
- Qt Centre Forum: <http://www.qtcentre.org/>
- Mailing lists: <http://lists.qt-project.org>
- IRC: irc.freenode.org channel: #qt
- Bug Report: <https://bugreports.qt.io>
- Tutorials: <http://wiki.qt.io/Developer-Guides>
- Video tutorials: www.youtube.com/user/QtStudios

Integrated Computer Solutions, Inc.
All Rights Reserved

Widgets vs Qt Quick (QML)

- Two ways of writing GUI
- The QWidget API is in C++, compiled and more suitable for Desktop Applications
- QML is a markup language which uses Javascript and is interpreted at run-time. More suitable for embedded devices.

Starting Your Project with Widgets

```
#include <QtWidgets>

int main(int argc, char *argv[])
{
    QApplication app(argc, argv);
    QPushButton button("Hello world"); // Creates a widget
    button.show();
    return app.exec(); // Starts the event loop
}
```

© Integrated Computer Solutions, Inc.
All Rights Reserved

Program consists of:

- main.cpp – application code
- helloworld.pro or CMakeLists.txt - project file



Starting Your Project with Qt Quick

Hello world

```
#include <QGuiApplication>
#include <QQmlApplicationEngine>

int main(int argc, char *argv[])
{
    QGuiApplication app(argc, argv);
    QQmlApplicationEngine engine;
    engine.load(QUrl(QStringLiteral("qrc:/main.qml")));

    return app.exec();
}
```

© Integrated Computer Solutions, Inc.
All Rights Reserved

Qt Quick - Hello World in Qt

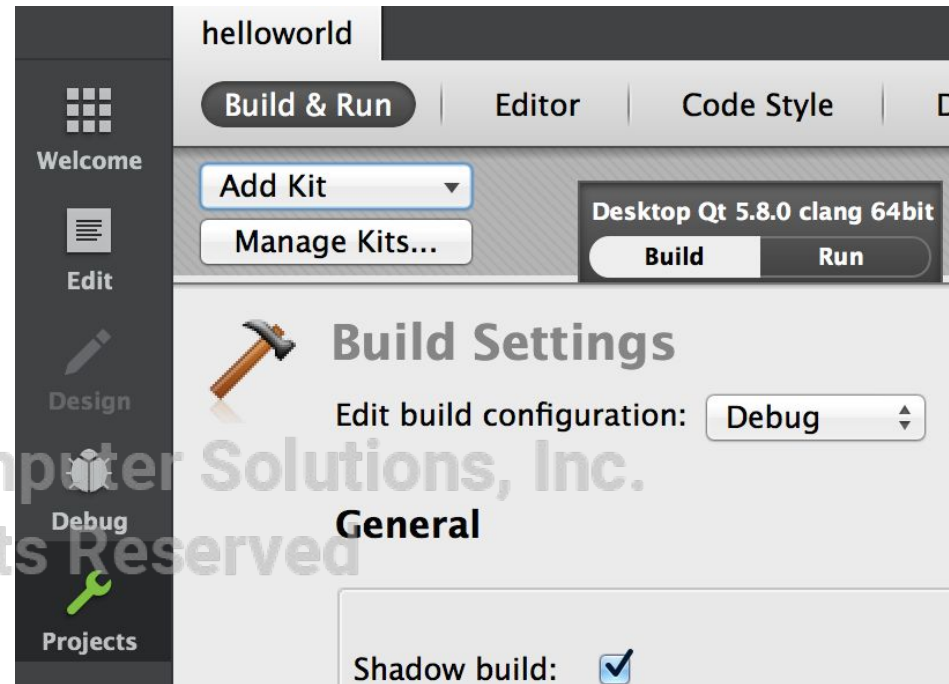
```
import QtQuick 2.4
import QtQuick.Window 2.2
```

```
Window {
    visible: true
    width: 360; height: 360
    Rectangle {
        anchors.fill: parent
        MouseArea {
            anchors.fill: parent
            onClicked: {
                Qt.quit()
            }
        }
    }
    Text {
        anchors.centerIn: parent
        text: "Hello World"
    }
}
}
```

Kits

A kit defines, which

- Qt version is used (Qt versions sheet – qmake location)
- compiler is used (Compilers sheet)
- debugger is used (Debuggers sheet)



Debug Screen

Debug - > Start Debugging (or F5)

The screenshot displays the Qt Creator IDE interface. The main window shows a C++ source file named `main.cpp` with the following code:

```
1 #include <QGuiApplication>
2 #include <QQmlApplicationEngine>
3
4
5 int main(int argc, char *argv[])
6 {
7     QGuiApplication app(argc, argv);
8
9     QQmlApplicationEngine engine;
10    const QUrl url(u"qrc:/helloWorld1/main.qml"_qs);
11    QObject::connect(&engine, &QQmlApplicationEngine::objectCreated,
12                    &app, [url](QObject *obj, const QUrl &objUrl) {
13        if (!obj && url == objUrl)
14            QApplication::exit(-1);
15    }, Qt::QueuedConnection);
16    engine.load(url);
17
18    return app.exec();
19 }
20
```

The variable inspector on the right shows the state of the `app` variable:

Name	Value
app	@0x7ffff...
[QCoreApplication]	@0x7ffff...
[d]	@0x5555...
[parent]	0x0
[children]	<2 items>
[properties]	<at least...
[methods]	<15 items>
[extra]	
staticMetaObject	@0x7ffff...
argc	1
argv	<1 items>

The debugger window at the bottom shows the current execution state:

Level	Function	File	Line	Address	Number	Function	File	Line	Address	Condition	Ignore
1	main	main...	9	0x55...	1	...r**)	...cpp	9	...6947		
2	...const	...cpp	15	...68c9							

The status bar at the bottom indicates the application is stopped at a breakpoint.

Qt Creator Shortcut Keys

Locator Prefixes

- F2 - Go to a symbol definition
- | <line number> - Go to a line in the current document
- ? <help topic> - Go to a help topic
- o <open document> - Go to an opened document
- <https://www.kdab.com/development-resources/qtcreator/>

Qt Books

- C++ GUI Programming with Qt 4 (2nd Edition) (Prentice Hall Open Source Software Development Series), Jasmin Blanchette and Mark Summerfield
- Practical Qt: Real World Solutions to Real World Problems, Matthias Kalle Dalheimer and Jesper Pederson
- An Introduction to Design Patterns in C++ with Qt 4, Alan Ezust and Paul Ezust

In the sessions

This five-part training will also cover:

- QVariant, QObject
- QML
- QWidgets
- Model/View

© Integrated Computer Solutions, Inc.
All Rights Reserved