Debugging Qt Quick Projects

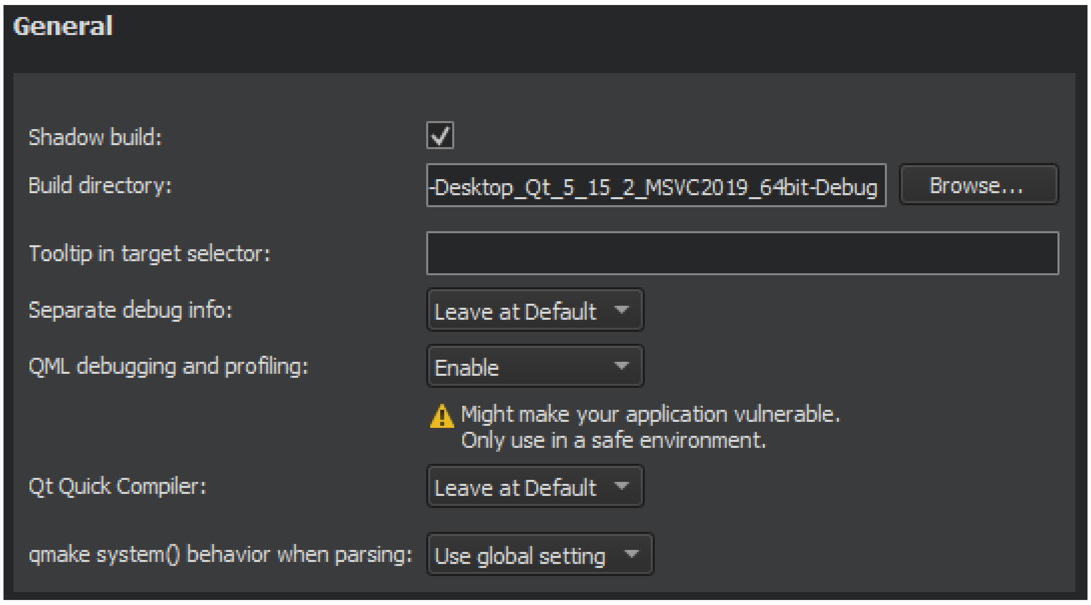
Note: You need Qt 5.0 or later to debug Qt Quick projects. For an example of how to debug Qt Quick Projects, see Debugging a Qt Quick Example Application.

Setting Up QML Debugging

The process of setting up debugging for Qt Quick projects depends on the type of the project: Qt Quick UI or Qt Quick Application, and the Qt version used. To debug Qt Quick UI projects, select the **Enable QML** check box in the **Debugger Settings** in **Projects** mode **Run Settings**.

To debug Qt Quick Applications:

1. If you use qmake as the build system, make sure that debugging is enabled in the **Build Settings**, **QML debugging and profiling** field, either explicitly for the project or globally by default.



General

Shadow build:	✓
Build directory:	-Desktop_Qt_5_15_2_MSVC2019_64bit-Debug Browse...
Tooltip in target selector:	
Separate debug info:	Leave at Default
QML debugging and profiling:	Enable
	⚠ Might make your application vulnerable. Only use in a safe environment.
Qt Quick Compiler:	Leave at Default
qmake system() behavior when parsing:	Use global setting

Note: Debugging requires opening a socket at a TCP port, which presents a security risk. Anyone on the Internet could connect to the application that you are debugging and execute any JavaScript functions. Therefore, you must make sure that the port is properly protected by a firewall.

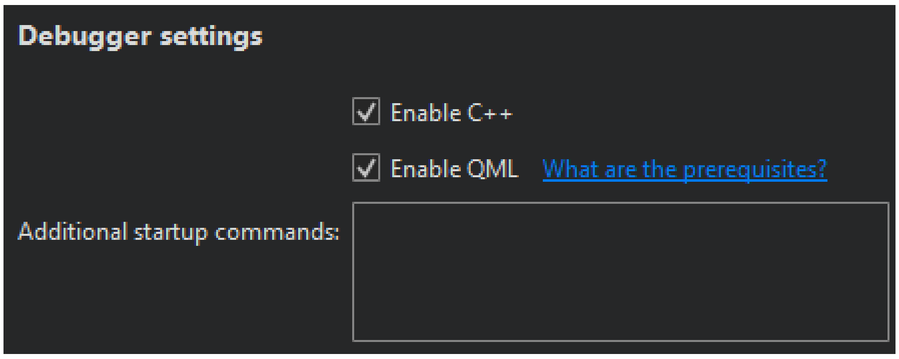
2. In the **Run Settings**, **Debugger Settings** section, select the **Enable QML** check box to enable QML debugging.
3. Select **Build** > **Rebuild Project** to clean and rebuild the project.

Qt DOCUMENTATION

Note: The `qmltooling` plugins that are required for debugging are automatically installed during Qt Creator and Qt installation. Do not delete them if you plan to debug QML applications.

Mixed C++/QML Debugging

To debug both the C++ and QML parts of your application at the same time, select the **Enable C++** and **Enable QML** checkboxes for both languages in the **Debugger Settings** section in the project **Run Settings**.



Starting QML Debugging

To start the application, choose **Debug** > **Start Debugging** > **Start Debugging of Startup Project** or press **F5**. Once the application starts running, it behaves and performs as usual. You can then perform the following tasks:

- Debug JavaScript functions
- Execute JavaScript expressions to get information about the state of the application
- Inspect QML properties and JavaScript variables and change them temporarily at runtime

To debug already running applications:

1. Build the application by using the appropriate configuration parameters (if you build the application with Qt Creator, it automatically uses the correct configuration):
   - When using CMake, the target_compile_definitions command is defined in the CMakeLists.txt file: `target_compile_definitions(myapp PRIVATE QT_QML_DEBUG)`

     Where *myapp* is the application to debug.
   - When using qmake, the following value is defined for the CONFIG property in the .pro file: `CONFIG += qml_debug`
2. Start the application with the following arguments:

   ```
   -qmljsdebugger=port:<port>[,host:<ip address>][,block]
   ```

   Where `port` (mandatory) specifies the debugging port, `ip address` (optional) specifies the IP address of the host where the application is running, and `block` (optional) prevents the application from running until the debug client connects to the server. This enables debugging from the start.

   Note: Setting breakpoints is only possible if the application is started with block mode.

Choose the kit configured for the device where the application to be debugged is running. The port number to use is displayed in the standard output when the application starts.

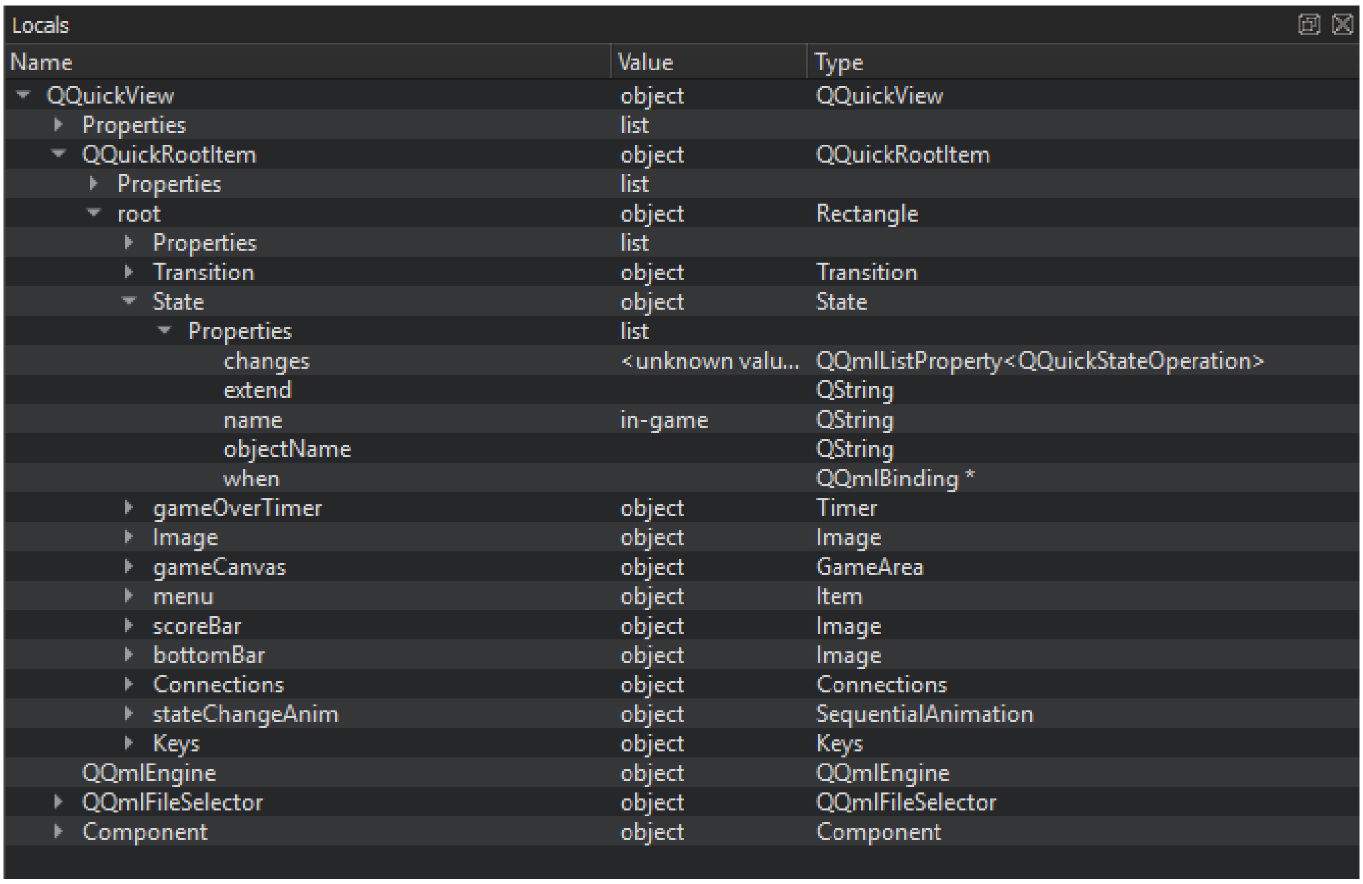
Debugging JavaScript Functions

You can use the Qt Creator **Debug** mode to inspect the state of your application while debugging. You can interact with the debugger by:

- Setting breakpoints
- Viewing call stack trace
- Viewing local variables and function parameters
- Evaluating Expressions

Inspecting Items

While the application is running, you can use the **Locals** view to explore the QML item structure.



```
Locals
Name                                Value              Type
QQuickView                          object             QQuickView
  Properties                        list
  QQuickRootItem                    object             QQuickRootItem
    Properties                      list
    root                            object             Rectangle
      Properties                    list
      Transition                    object             Transition
      State                         object             State
        Properties                  list
          changes                   <unknown valu...   QQmlListProperty<QQuickStateOperation>
          extend                                       QString
          name                      in-game            QString
          objectName                                   QString
          when                                         QQmlBinding *
      gameOverTimer                 object             Timer
      Image                         object             Image
      gameCanvas                    object             GameArea
      menu                          object             Item
      scoreBar                      object             Image
      bottomBar                     object             Image
      Connections                   object             Connections
      stateChangeAnim               object             SequentialAnimation
      Keys                          object             Keys
  QQmlEngine                        object             QQmlEngine
  QQmlFileSelector                  object             QQmlFileSelector
  Component                         object             Component
```

To keep the application visible while you interact with the debugger, select **Debug** > **Show Application on Top**.

You can view a QML item in the **Locals** view in the following ways:

- Expand the item in the object tree.
- Select the item in the code editor.
- Select **Debug** > **Select** to activate selection mode and then click an item in the running application.

Inspecting User Interfaces

When you debug complex applications, you can jump to the position in code where an item is defined.

In the selection mode, you can click items in the running application to jump to their definitions in the code. The properties of the selected item are displayed in the **Locals** view.

The **Select** tool will be enabled either if your application is using Qt 5.7 or later, or if your application is using an earlier version of Qt and is based on the `QQuickView` class. You can also view the item hierarchy in the running application:

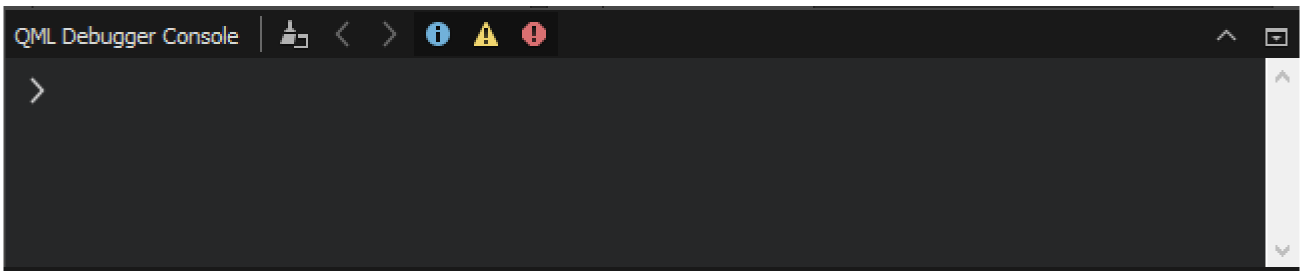
Double-click an item in the running application to cycle through the item stack at the cursor position.

To switch out of the selection mode, toggle the **Select** menu item.

To move the application running in Qt QML Viewer to the front, select **Debug** > **Show Application on Top**.

Executing JavaScript Expressions

When the application is interrupted by a breakpoint, you can use the **QML Debugger Console** to execute JavaScript expressions in the current context. To open it, choose **View** > **Output** > **QML Debugger Console**.



You can change property values temporarily, without editing the source, and view the results in the running application. You can change the property values permanently in code.

Applying QML Changes at Runtime

When you change property values in the **QML Debugger Console** or in the **Locals** or **Expression** view, they are immediately updated in the running application, but not in the source code.

‹ Using Debugging Helpers Debugging a C++ Example Application ›

Qt DOCUMENTATION

Contact Us

Company

About Us
Investors
Newsroom
Careers
Office Locations

Licensing

Terms & Conditions
Open Source
FAQ

Support

Support Services
Professional Services
Partners
Training

For Customers

Support Center
Downloads
Qt Login
Contact Us
Customer Success

Community

Contribute to Qt
Forum
Wiki
Downloads
Marketplace

Feedback Sign In