调试Qt快速项目

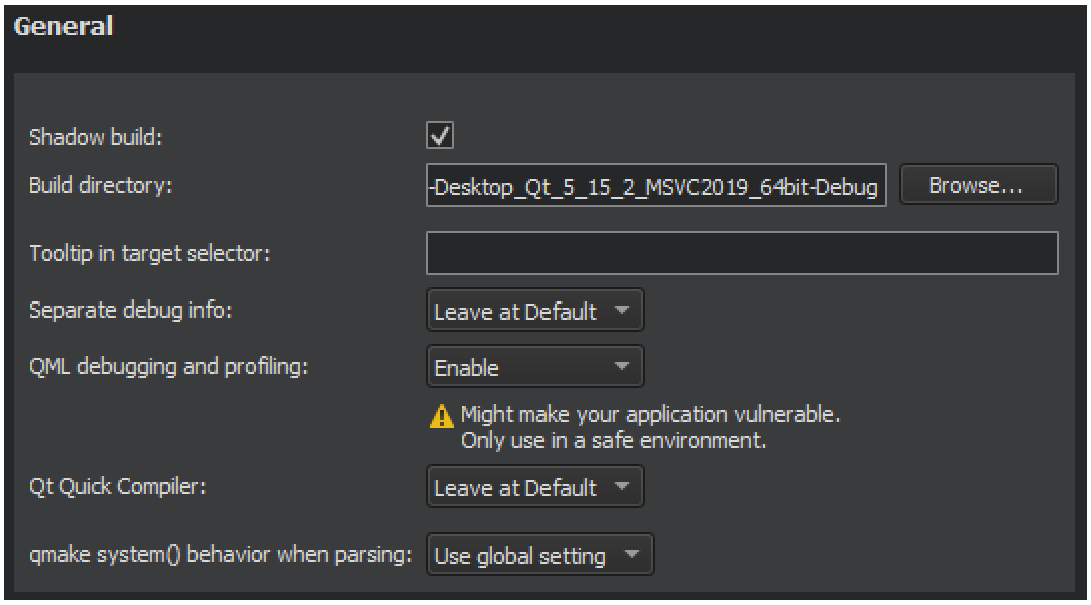
注意： 您需要 Qt 5.0 或更高版本来调试 Qt 快速项目。有关如何调试 Qt 快速项目的示例，请参见调试 Qt 快速示例应用程序。

设置 QML 调试

为 Qt 快速项目设置调试的过程取决于项目的类型：Qt 快速 UI 或 Qt 快速应用程序，以及使用的 Qt 版本。若要调试 Qt 快速 UI 项目，请在"**项目模式的调试器设置**"中选中"**启用 QML"复选框 运行设置。**

调试Qt快速应用程序：

1. 如果使用 qmake 作为生成系统，请确保在"**生成设置**"、"**QML 调试和分析**"字段中启用调试，可以是显式为项目启用的，也可以是默认的全局调试。



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

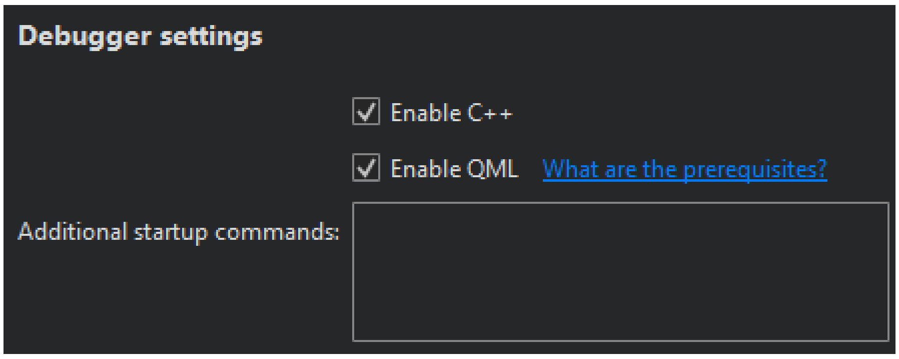
注意： 调试需要在 TCP 端口处打开套接字，这会带来安全风险。互联网上的任何人都可以连接到您正在调试的应用程序并执行任何 JavaScript 函数。因此，必须确保端口受到防火墙的适当保护。

2. 在"**运行设置**"、"**调试器设置**"部分，选中"**启用 QML**"复选框以启用 QML 调试。
3. 选择"**生成**>**重新生成项目**"以清理并重新生成项目。
4. 若要在设备上调试应用程序，请检查设备上是否安装了 Qt 5.0 或更高版本的库，并在开始调试之前为设备选择相应的工具包

除它们。qmltooling

混合C++/QML 调试

若要同时调试应用程序的 C++ 和 QML 部分，请在项目的“**运行设置”的“调试器设置**”部分中为这两种语言选中“**启用C++**”和“**启用 QML**”复选框。


Debugger settings

☑ Enable C++

☑ Enable QML What are the prerequisites?

Additional startup commands:

启动 QML 调试

若要启动应用程序，请选择“**调试>”启动调试**“>**启动项目的”启动调试**“或按 F5。应用程序开始运行后，其行为和执行将照常进行。然后，您可以执行以下任务：

- 调试脚本函数
- 执行脚本表达式以获取有关应用程序状态的信息
- 检查 QML 属性和脚本变量，并在运行时临时更改它们

调试已在运行的应用程序：

1. 使用适当的配置参数构建应用程序（如果使用 Qt Creator 构建应用程序，它会自动使用正确的配置）：
   - 使用 CMake 时，target_compile_definitions命令在“.txt”文件中定义：`target_compile_definitions(myapp PRIVATE QT_QML_DEBUG)`

     其中 *myapp* 是要调试的应用程序。
   - 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 (mandatory) specifies the debugging port, (optional) specifies the IP address of the host where the application is running, and (optional) prevents the application from running until the debug client connects to the server. This enables debugging from the start.`port` `ip address` `block`

   > **Note:** Setting breakpoints is only possible if the application is started with block mode.
3. Select **Debug** > **Start Debugging** > **Attach to QML Port**.

Qt DOCUMENTATION

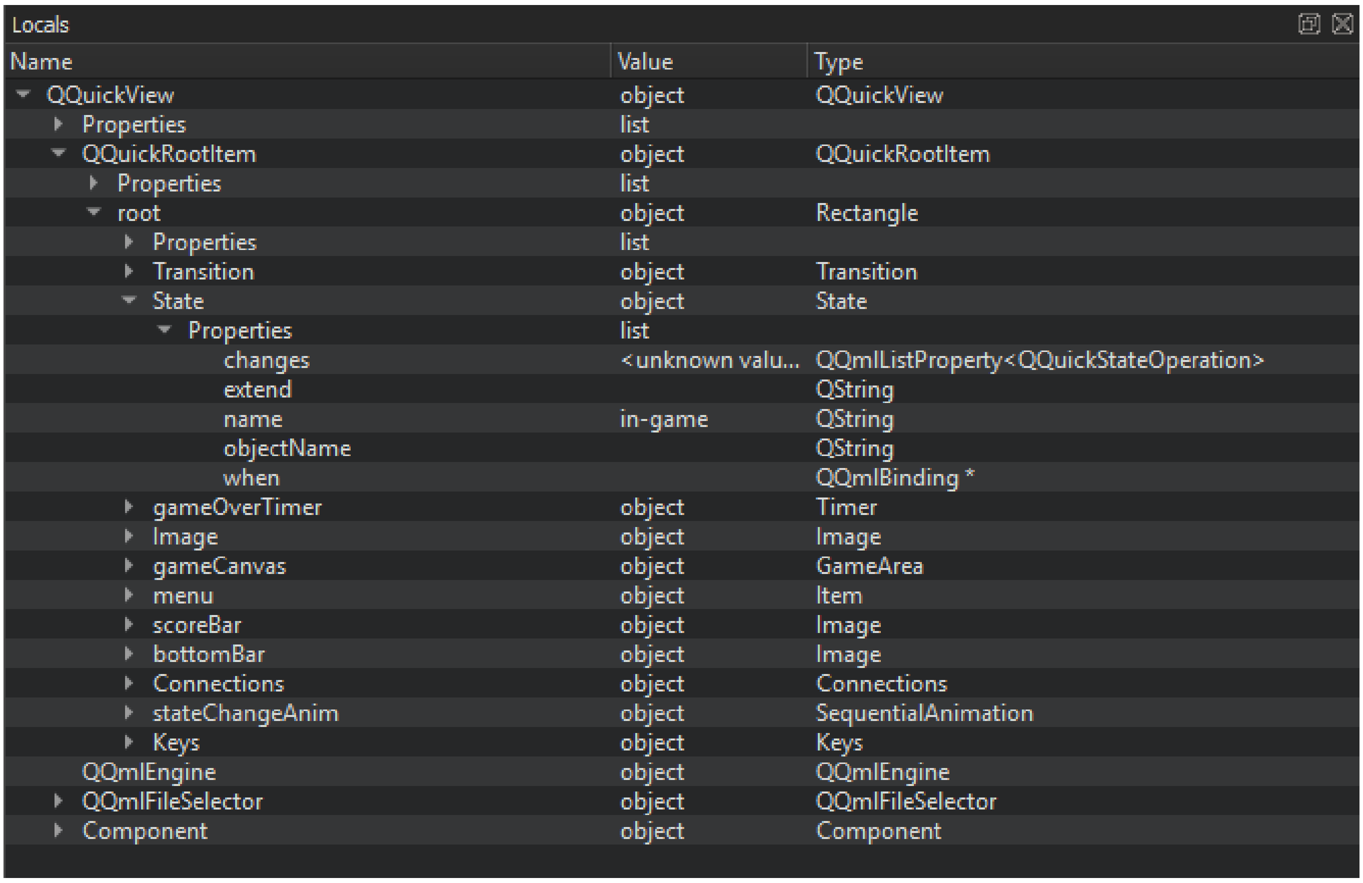
Debugging JavaScript Functions

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.

To change property values temporarily, without editing the source, double-click them and enter the new values. You can view the results in the running application.

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 class. You can also view the item hierarchy in the running application:`QQuickView`

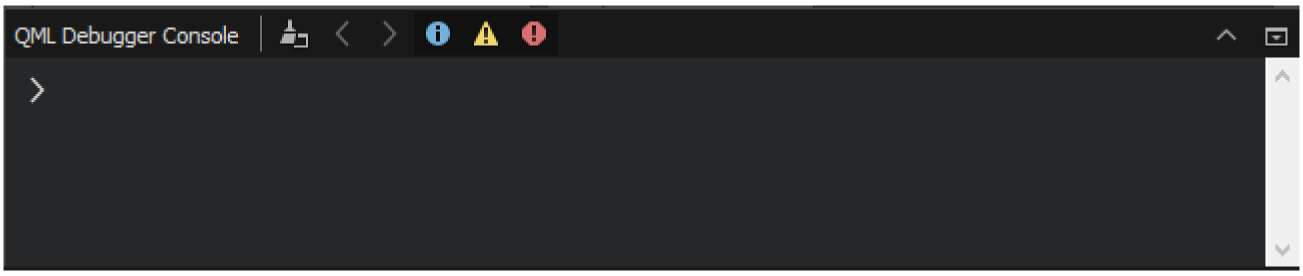
Double-click an item in the running application to cycle through the item stack at the cursor position.

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

The Qt Company

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