

[Qt设计工作室手册](#) > [调试Qt快速项目](#)

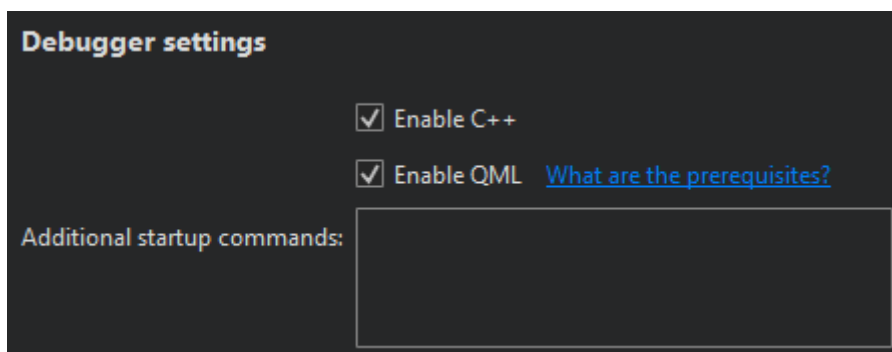
# 调试Qt快速项目

有关如何调试Qt快速项目的示例，请参阅[调试Qt快速示例应用程序](#)。

**注意：**在本节中，您将使用高级菜单项。默认情况下，这些是不可见的。要切换高级菜单项的可见性，请参阅[自定义菜单](#)。

## 设置 QML 调试

要调试Qt Quick UI项目，请在**项目模式运行**设置的**调试器设置**中选中启用QML复选框。



## 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 JavaScript 函数
- › 执行 JavaScript 表达式以获取有关应用程序状态的信息
- › 检查 QML 属性和 JavaScript 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. 使用以下参数启动应用程序：

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

**注意：**仅当应用程序以块模式启动时，才能设置断点。

2. 选择**调试>启动调试>附加到 QML 端口**。

选择为运行要调试的应用程序的设备配置的工具包。应用程序启动时，要使用的端口号将显示在标准输出中。

## 调试 JavaScript 函数

您可以使用 Qt 设计工作室调试模式在**调试**时检查应用程序的状态。可以通过以下方式与调试器交互：

- › 设置断点
- › 查看调用堆栈跟踪
- › 查看局部变量和函数参数
- › 计算表达式


## 设置断点


可以将断点与以下项相关联：

- › 源代码文件和行
- › 功能
- › 地址
- › 引发和捕获异常
- › 执行和分叉进程
- › 执行某些系统调用
- › 程序运行时特定地址的内存块中的更改
- › 发射 QML 信号
- › 抛出 JavaScript 异常

在某些条件下，可以限制断点对程序的中断。

断点有两种类型：和。未认领的断点表示中断调试程序的任务，并在以后将控制权传递给您。它有两种状态：和。unclaimedclaimedpendingimplanted

未声明的断点存储为会话的一部分，并且独立于程序是否正在调试而存在。当它们引用代码中的位置时，它们会在“断点**预设**”视图和编辑器中使用（**未认领的断点**）图标列出。

Breakpoint Preset							
Debuggee	Function	File	Line	Address	Condition	Ignore	Threads
	-	...\quickcontrols2\gallery\gallery.qml	155				(all)

当调试器启动时，调试器会列出可能由调试程序处理的代码。认领断点由红色断点图标表示，并会阻止此断点以允许调试。

在不同时间，会尝试将挂起的断点植入调试的进程。成功完全植入可能会创建一个或多个植入断点，每个断点都与调试断点中的实际地址相关联。例如，植入还可能将编辑器中的断点标记从空行移动到为其生成实际代码的下一行。植入的断点图标没有沙漏覆盖。

调试器结束时，其声明的断点（挂起和植入）将返回到未声明状态，并重新出现在“断点预设”视图中。

如果在执行调试的程序期间命中植入的断点，控制权将传递回给您。然后，您可以检查中断程序的状态，或者逐行或连续继续执行。

Number	Function	File	Line	Address	Condition	Ignore	Threads
2	-	f:\dd\vc\tools\crt\vcstartup\src\startup\exe_main.cpp	17	0x7ff7447b9e3d			(all)

## 添加断点

添加断点：

- 通过以下方式之一添加新断点：
  - 在代码编辑器中，单击左边距或在您希望程序停止的特定行上按F9（在 macOS 上为 F8）。
  - 在“断点预设”视图或“断点”视图中：
    - 双击视图的空白部分。
    - 右键单击视图，然后在上下文菜单中选择“添加断点”。
- 在“断点类型”字段中，选择程序代码中希望程序停止的位置。要指定的其他选项取决于所选位置。

Qt Add Breakpoint

Basic

Breakpoint type: File Name and Line Number

File name:

Browse...

Line number:

0

Enabled:

☒

Address:

Expression:

Function:

One shot only:

☐

Advanced

Condition:

Commands:

Ignore count:

0

Thread specification:

(all)

Path:

Use Engine Default

Module:

3. 在“条件”字段中，如果**条件**的计算结果为 true，则设置在断点处停止之前要评估的条件。
4. 在“**忽略**”字段中，指定在程序停止之前忽略断点的次数。
5. 在“命令”字段中，指定程序停止时要执行的**命令**；一行上的一个命令。GDB 按指定的顺序执行命令。

## 移动断点

移动断点：

- › 将断点标记拖放到文本编辑器中的另一行。
- › 在“断点预设”视图或“断点”视图中，选择“**编辑所选断点**”，然后在“行号”字段中设置**行号**。

## 删除断点

删除断点：

- › 单击文本编辑器中的断点标记。
- › 在“断点预设”视图或“**断点**”视图中：
  - › 选择断点，然后按Delete 键。
  - › 在上下文菜单中选择“删除选定的断点”、“删除**选定的断点**”或“删除文件的断点”。

## 启用和禁用断点

要暂时禁用断点而不删除断点并丢失条件和命令等关联数据，请执行以下操作：

- › 右键单击文本编辑器中的断点标记，然后选择“**禁用断点**”。
- › 选择包含断点的行，然后按 Ctrl+F9（在 macOS 上按 Ctrl+F8）。
- › 在“断点预设”视图或“**断点**”视图中：
  - › 选择断点，然后按**空格键**。
  - › 在上下文菜单中选择“**禁用断点**”。

文本编辑器和视图中的空心断点图标表示禁用的断点。若要重新启用断点，请使用上述任一方法。

除了数据断点之外，断点在重新启动调试的程序时将保持其启用或禁用状态。

## 设置数据断点

在指定地址读取或写入数据时，**数据断点**将停止程序。

在地址处设置数据断点：

1. 在“断点预设”或“断点”视图中，选择上下文菜单中的“**添加断点**”。
2. 在**断点类型**字段中，选择**在固定地址访问数据时中断**。
3. 在地址字段中，指定内存块的**地址**。
4. 选择“**确定**”。

如果地址显示在“**局部变量**”或“**表达式**”视图中，则可以在上下文菜单中选择“**在对象的地址处添加数据断点**”以设置数据断点。

## 查看调用堆栈跟踪

当正在调试的程序中断时，Qt Design Studio将显示指向当前位置的嵌套函数调用作为调用堆栈跟踪。此堆栈跟踪是从调用堆栈帧构建的，每个调用堆栈帧表示一个特定的函数。对于每个函数，Qt Design Studio都会尝试检索相应源文件的文件名和行号。此数据显示在**堆栈**视图中。

Stack			
Level	Function	File	Line
→ 1	TextFinder::loadTextFile	textfinder.cpp	85
2	TextFinder::TextFinder	textfinder.cpp	64
3	main	main.cpp	59
4	invoke_main	exe_common.inl	79
5	__scrt_common_main_seh	exe_common.inl	283
6	__scrt_common_main	exe_common.inl	326
7	mainCRTStartup	exe_main.cpp	17
8	BaseThreadInitThunk	KERNEL32	
9	RtlUserThreadStart	ntdll	

由于导致当前位置的调用堆栈可能产生或经过没有调试信息的代码，因此并非所有堆栈帧都具有相应的源位置。没有相应源位置的堆栈帧在**“堆栈”**视图中灰显。

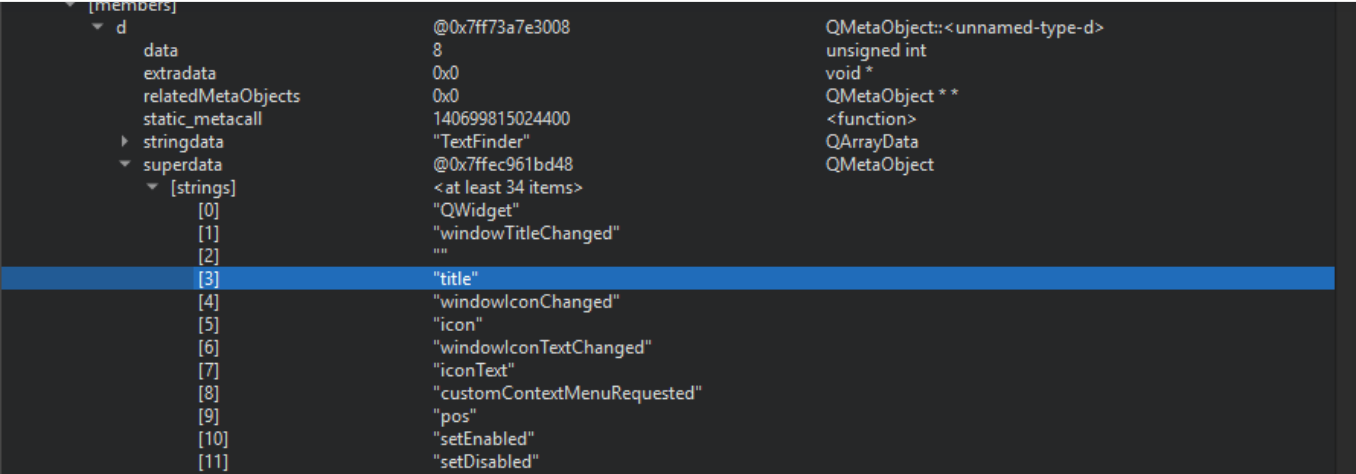
如果单击具有已知源位置的帧，文本编辑器将跳转到相应的位置并更新**“局部变量”**和**“表达式”**视图，使程序在进入函数之前似乎被中断了。

要找出哪个 QML 文件导致 Qt Quick 2 应用程序崩溃，请在堆栈视图的上下文菜单中选择**加载 QML 堆栈**。调试器尝试从停止的可执行文件中检索 JavaScript 堆栈，并将帧附加到 C++ 帧（如果找到任何帧）。您可以单击 QML 堆栈中的帧以在编辑器中打开 QML 文件。

## 局部变量和函数参数

**“局部变量”**视图由**“局部变量”**窗格和**“返回值”**窗格（空时隐藏）组成。

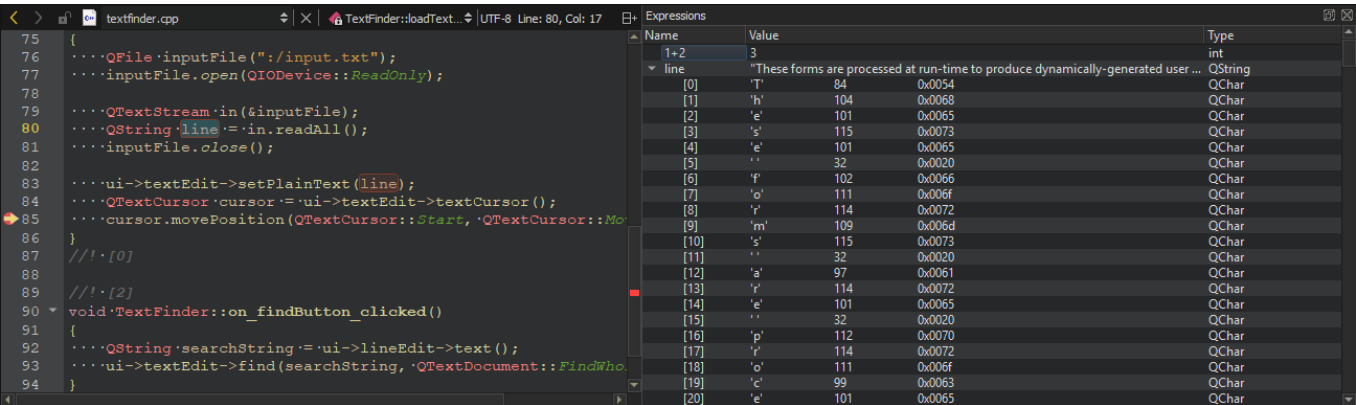
Locals		
Name	Value	Type
▸ a	@0xc8b7fdc0	QApplication
argc	<not accessible>	int
argv	<0 items>	char **
▾ w	@0xc8b7fd88	TextFinder
▸ [QWidget]	@0xc8b7fd88	QWidget
▾ staticMetaObject	@0x7ff73a7e3008	QMetaObject
▸ [strings]	<at least 1 items>	



使用 GDB 时，可以指定是显示对象的动态类型还是静态类型。在上下文菜单中选择**使用动态对象类型进行显示**。请记住，选择动态类型可能会较慢。

## 计算表达式

要计算算术表达式或函数调用的值，请使用表达式视图中的**表达式**计算器。要插入新的表达式计算器，请双击“表达式”或“**局部变量**”视图的空白部分，或者从上下文菜单中选择“**添加新表达式计算器**”，或者从代码编辑器中拖放**表达式**。



**注意：**表达式计算器功能强大，但会显著减慢调试器操作的速度。建议不要过度使用它们，并尽快删除不需要的表达式计算器。

每当当前帧更改时，都会重新计算表达式计算器。请注意，每次都会调用表达式中使用的函数，即使它们有副作用。

QML 调试器可以评估 JavaScript 表达式。

## 检查项目

在应用程序运行时，您可以使用“**局部变量**”视图浏览 QML 项目结构。

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

若要在与调试器交互时使应用程序保持可见，请选择“**调试**>在顶部显示应用程序”。

您可以通过以下方式在“**局部变量**”视图中查看 QML 项目：

- › 展开对象树中的项。
- › 在代码编辑器中选择该项。
- › 选择“**调试**”>**选择**以激活选择模式，然后单击正在运行的应用程序中的某个项目。

若要临时更改属性值而不编辑源，请双击它们并输入新值。您可以在正在运行的应用程序中查看结果。

## 检查用户界面

调试复杂应用程序时，可以跳转到代码中定义项的位置。

在选择模式下，可以单击正在运行的应用程序中的项以跳转到代码中的项定义。所选项的属性将显示在“**局部变量**”视图中。

您还可以在正在运行的应用程序中查看项目层次结构：

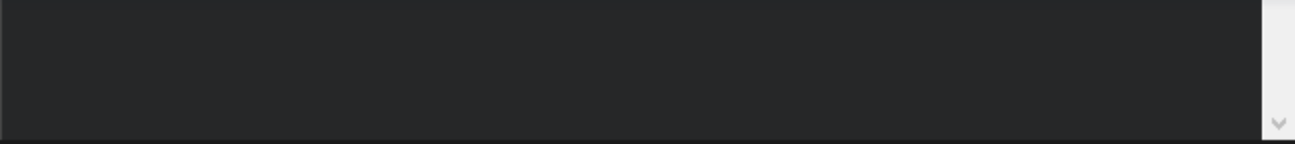
双击正在运行的应用程序中的某个项，以在光标位置循环浏览项堆栈。

要切换出选择模式，请切换“**选择**”菜单项。

要将 Qt QML 查看器中运行的应用程序移到前面，请选择**调试**>在顶部显示应用程序。

## 执行 JavaScript 表达式

当应用程序被断点中断时，您可以使用**QML 调试器控制台**在当前上下文中执行 JavaScript 表达式。若要打开它，请选择“**视图**>**输出**>**QML 调试器控制台**”。



可以临时更改属性值，而无需编辑源，并在正在运行的应用程序中查看结果。可以在“属性”视图中永久更改属性值。

## 在运行时应用 QML 更改

在QML 调试器控制台或“局部变量”或“表达式”视图中更改属性值时，这些属性值会立即在正在运行的应用程序中更新，但不会在源代码中更新。

[◀ 调试和分析](#)

[调试Qt快速示例应用程序 ▶](#)



联系我们

### 公司

- 关于我们
- 投资者
- 编辑部
- 职业
- 办公地点

### 发牌

- 条款和条件
- 开源
- 常见问题

### 支持

- 支持服务
- 专业服务
- 合作伙伴
- 训练

### 对于客户

- 支持中心
- 下载
- Qt登录
- 联系我们
- 客户成功案例

### 社区





维基  
下载  
市场

© 2022 Qt公司

反馈 登录