Debugging a Qt Quick Example Application

This section uses the Same Game example application to illustrate how to debug Qt Quick applications in the **Debug** mode.

For more information about all the options you have, see Debugging Qt Quick Projects.

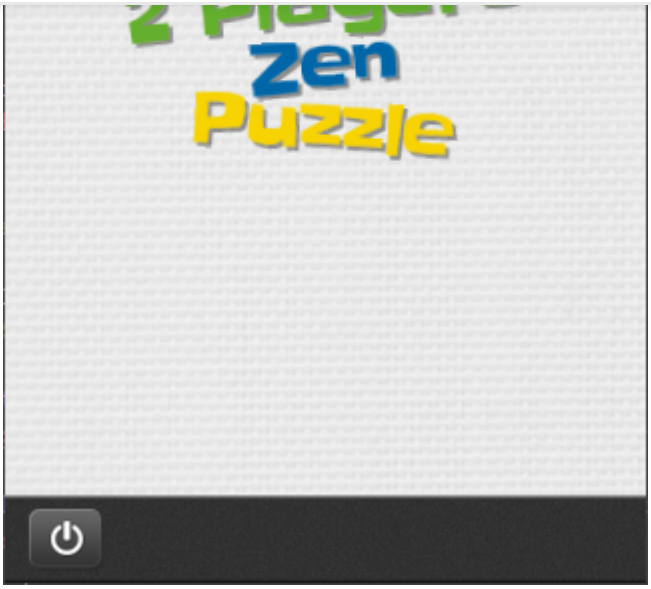
> **Note:** In this section, you are using advanced menu items. These are not visible by default. To toggle the visibility of advanced menu items, see Customizing the Menu.

The Same Game demo shows how to write a game in QML, using JavaScript for all the game logic. Open the demo project in Qt Design Studio to debug it:

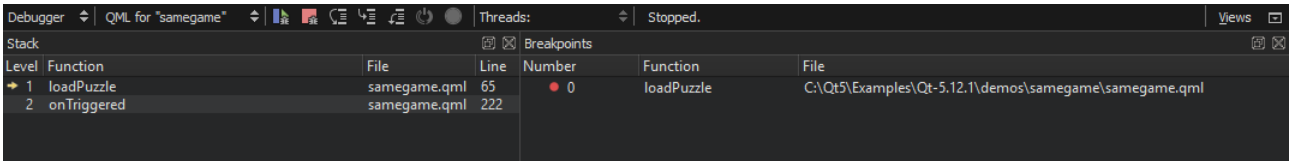
1. To look at the code that starts a new game, place a breakpoint in samegame.qml by clicking between the line number and the window border on the line where where the `startNewGame()` function is called (1).

   The red circle indicates that a breakpoint is now set on that line number.

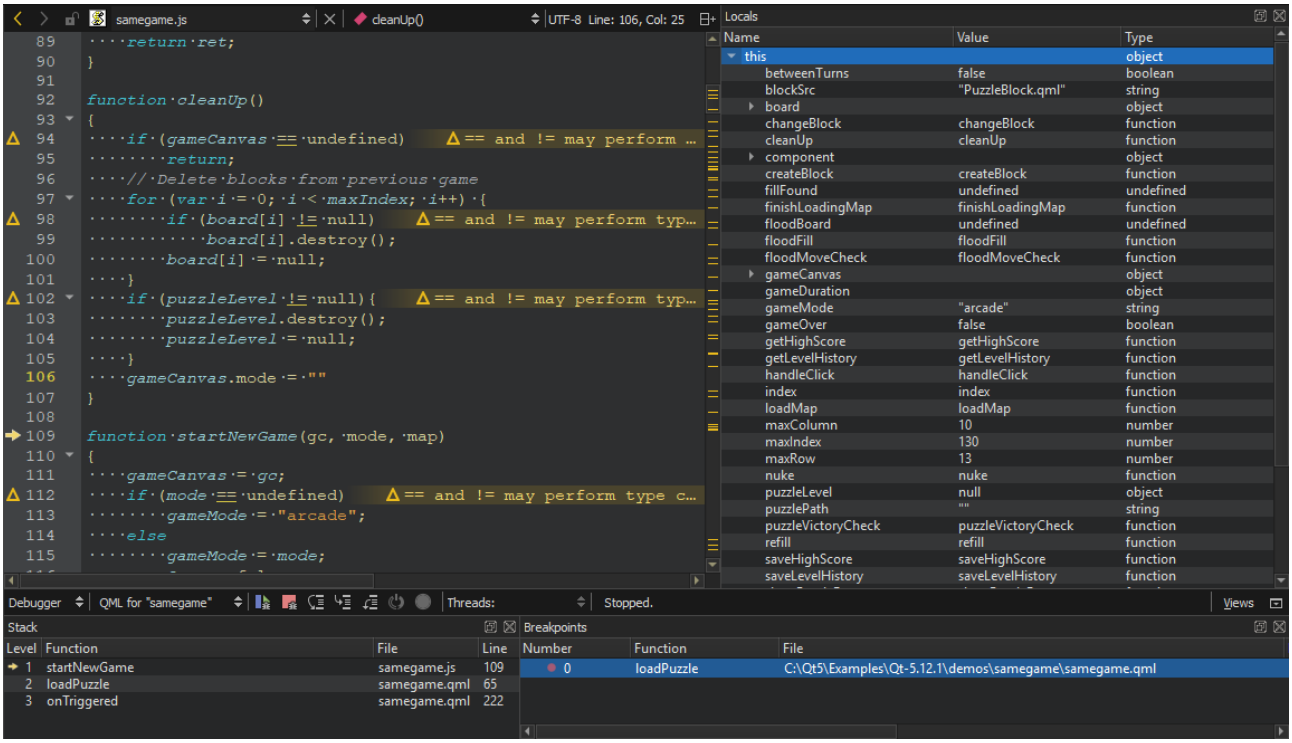
2. Select **Debug** > **Start Debugging** > **Start Debugging of Startup Project** or press **F5**.
3. Once the Same Game application starts, select **Puzzle** to start a new game.



4. When the debugger hits the breakpoint, it interrupts the application. Qt Design Studio displays the nested function calls leading to the current position as a call stack trace (1).

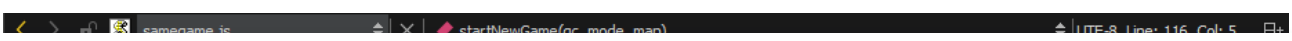


5. Click the (**Step Into**) button on the toolbar or press **F11** to step into the code in the stack. The samegame.js file opens in the code editor at the function that starts a new game.



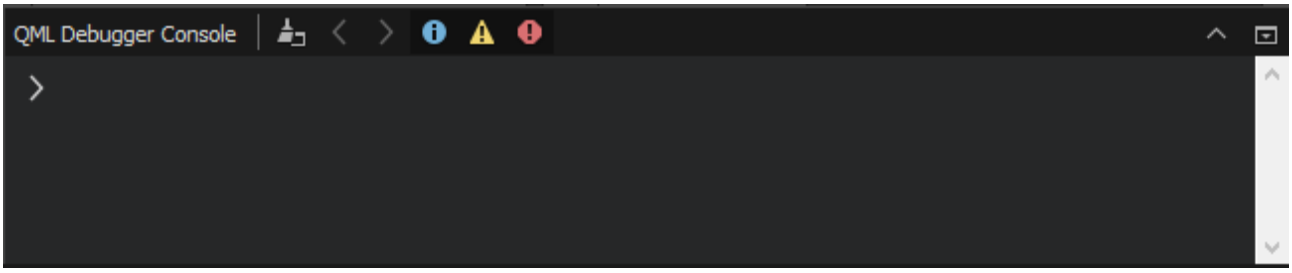
6. Examine the local variables in the **Locals** view. Step through the code to see how the information changes in the view.

7. Add a breakpoint at the end of the `startNewGame()` function, and click (**Continue**) to hit the breakpoint.

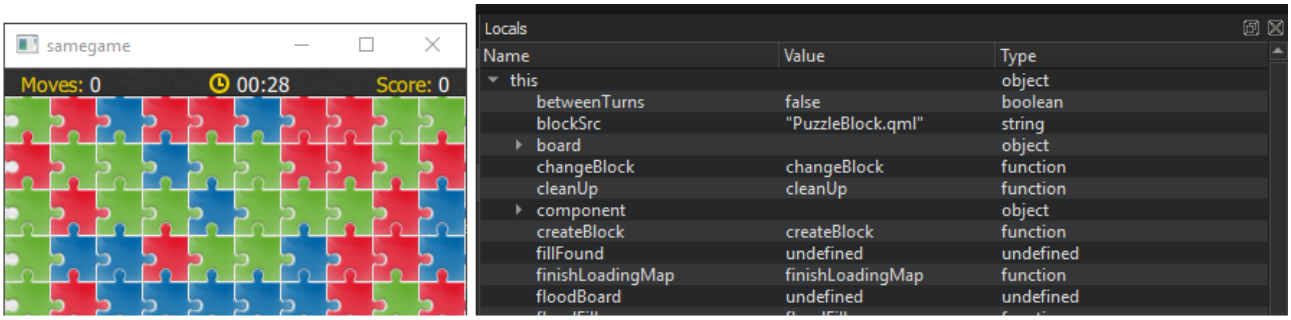


```
130     ····//·Initialize·Board
131     ····board·=·new·Array(maxIndex);
132     ····gameCanvas.score·=·0;
133     ····gameCanvas.score2·=·0;
134     ····gameCanvas.moves·=·0;
135     ····gameCanvas.curTurn·=·1;
136     ····if·(gameMode·==·"puzzle")    ⚠ == and != may perform type coercion, use === or !== to av…
137     ········loadMap(map);
138     ····else//Note·that·we·load·them·in·reverse·order·for·correct·visual·stacking
139     ········for·(var·column·=·maxColumn·-·1;·column·>=·0;·column--)
140     ············for·(var·row·=·maxRow·-·1;·row·>=·0;·row--)
141     ················createBlock(column,·row);
142     ····if·(gameMode·==·"puzzle")    ⚠ == and != may perform type coercion, use === or !== to av…
143     ········getLevelHistory();//Needs·to·be·after·map·load
144     ····gameDuration·=·new·Date();
145     }
```

8. To execute JavaScript commands in the current context, open the **QML Debugger Console**.



9. To remove a breakpoint, right-click it and select **Delete Breakpoint**.
10. In the **Locals** view, explore the object structure at runtime.



11. Select **Debug** > **Show Application on Top** to keep the application visible while you interact with the debugger.
12. Select **Debug** > **Select** to activate selection mode and then click the **Menu** button to move into the **menuButton** component in the **Locals** view and the code editor.
13. In the **Locals** view, double-click the value of a property to change it.

‹ Debugging Qt Quick Projects Profiling QML Applications ›

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