Qt DOCUMENTATION

Search

Qt Creator Manual 8.0.2

Topics >

Qt Creator Manual > Analyzing Code

You can use the code analysis tools in the **Debug** mode. To switch to **Debug** mode, select **Debug** in the mode selector, or select the **Analyze** menu and then select a tool. When you are in the **Debug** mode, you can switch between tools by selecting them in the menu on the toolbar.

You can drag and drop the views in the **Debug** mode to new positions on the screen. The size and position of views are saved for future sessions. Select **View** > **Views** > **Reset to Default Layout** to reset the views to their original sizes and positions.

You can use the following code analysis tools in the **Debug** mode:

- QML Profiler

  Inspect binding evaluations, signal handling, and painting operations when running QML code. This is useful for identifying potential bottlenecks, especially in the evaluation of bindings.

- Coco

  Analyze the way an application runs as part of a test suite, for example, and use the results to make the tests more efficient and complete.

- Valgrind Code Analysis Tools

  Detect problems in memory management by using the Memcheck tool and find cache misses in the code by using the Callgrind tool.

- Clang Tools

  Detect problems in C, C++, and Objective-C programs by using Clang-Tidy and Clazy.

- Heob

  Use the Heob heap observer on Windows to detect buffer overruns and memory leaks.

- Performance Analyzer

  Analyze the CPU usage of embedded applications and Linux desktop applications with the Performance Analyzer that integrates the Linux Perf tool.

- Cppcheck

  Use the experimental Cppcheck plugin to detect undefined behavior and dangerous coding constructs.

- Chrome Trace Format Visualizer

  Use the Chrome Trace Format (CTF) Visualizer to view Chrome trace events. This is especially useful when viewing large trace files that are difficult to visualize using the built-in trace-viewer (`chrome://tracing`).

< Troubleshooting Debugger   Profiling QML Applications >