Using the debugger

Contents

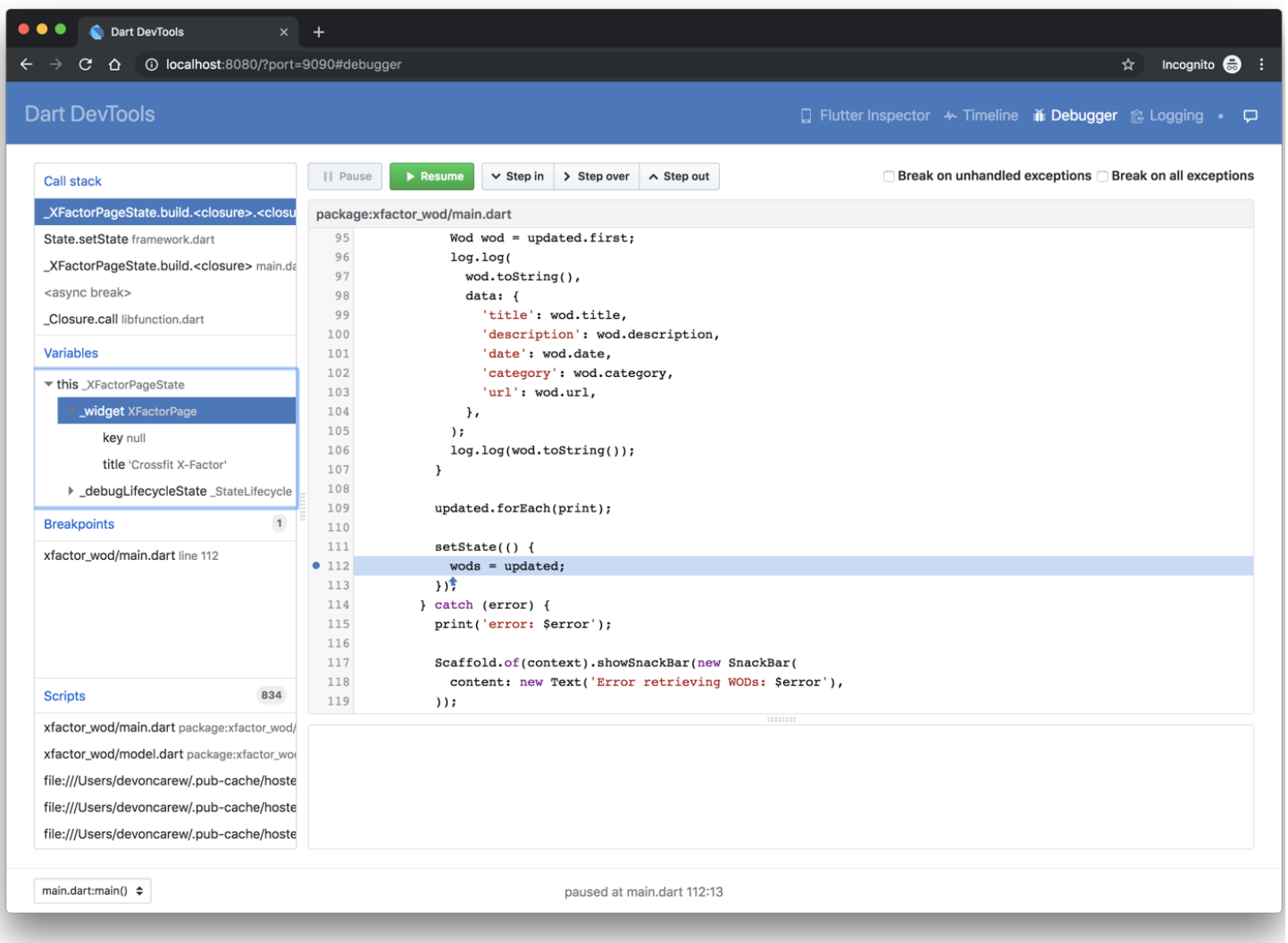
Note: The debugger works with Flutter mobile and web applications.

Getting started

DevTools includes a full source-level debugger, supporting breakpoints, stepping, and variable inspection.

When you open the debugger tab, you should see all the libraries for your application listed in the bottom left screen (under the **Scripts** area), and the source for the main entry-point for your app in is loaded in the main app source area.

In order to browse around more of your application sources, you can scroll through the `Scripts` area and select other source files display.



Setting breakpoints

To set a breakpoint, click the left margin (the line number ruler) in the source area. Clicking once sets a breakpoint, which should show up in the **Breakpoints** area on the left. Clicking again removes the breakpoint.

The call stack and variable areas

When your application encounters a breakpoint, it pauses there, and the DevTools debugger shows the paused execution location the source area. In addition, the `Call stack` and `Variables` areas populate with the current call stack for the paused isolate, and local variables for the selected frame. Selecting other frames in the `Call stack` area changes the contents of the variables.

Within the `Variables` area, you can inspect individual objects by toggling them open to see their fields. Hovering over an object in `Variables` area calls `toString()` for that object and displays the result.

Stepping through source code

When paused, the three stepping buttons become active.

- Use **Step in** to step into a method invocation, stopping at the first executable line in that invoked method.
- Use **Step over** to step over a method invocation; this steps through source lines in the current method.
- Use **Step out** to step out of the current method, without stopping at any intermediary lines.

In addition, the **Resume** button continues regular execution of the application.

Console output

Console output for the running app (stdout and stderr) is displayed in the console, below the source code area. You can also see output in the Logging view.

Breaking on exceptions

To adjust the break-on-exceptions behavior, toggle the **Break on unhandled exceptions** and **Break on all exceptions** checkboxes in the upper right of the debugger view.

Breaking on unhandled excepts only pauses execution if the breakpoint is considered uncaught by the application code. Breaking all exceptions causes the debugger to pause whether or not the breakpoint was caught by application code.

Known issues

When performing a hot restart for a Flutter application, user breakpoints are cleared.

Other resources

For more information on debugging and profiling, see the Debugging page.