Debugging on Windows

If you experience crashes or issues in Electron that you believe are not caused by your JavaScript application, but instead by Electron itself, debugging can be a little bit tricky, especially for developers not used to native/C++ debugging. However, using Visual Studio, Electron's hosted Symbol Server, and the Electron source code, you can enable step-through debugging with breakpoints inside Electron's source code.

See also: There's a wealth of information on debugging Chromium, much of which also applies to Electron, on the Chromium developers site: Debugging Chromium on Windows.

Requirements

- **A debug build of Electron**: The easiest way is usually building it yourself, using the tools and prerequisites listed in the build instructions for Windows. While you can attach to and debug Electron as you can download it directly, you will find that it is heavily optimized, making debugging substantially more difficult: The debugger will not be able to show you the content of all variables and the execution path can seem strange because of inlining, tail calls, and other compiler optimizations.
- **Visual Studio with C++ Tools**: The free community editions of Visual Studio 2013 and Visual Studio 2015 both work. Once installed, configure Visual Studio to use Electron's Symbol server. It will enable Visual Studio to gain a better understanding of what happens inside Electron, making it easier to present variables in a human-readable format.
- **ProcMon**: The free SysInternals tool allows you to inspect a processes parameters, file handles, and registry operations.

Attaching to and Debugging Electron

To start a debugging session, open up PowerShell/CMD and execute your debug build of Electron, using the application to open as a parameter.

```
$ ./out/Testing/electron.exe ~/my-electron-app/
```

Setting Breakpoints

Then, open up Visual Studio. Electron is not built with Visual Studio and hence does not contain a project file - you can however open up the source code files "As File", meaning that Visual Studio will open them up by themselves. You can still set breakpoints - Visual Studio will automatically figure out that the source code matches the code running in the attached process and break accordingly.

Relevant code files can be found in `./shell/`.

Attaching

You can attach the Visual Studio debugger to a running process on a local or remote computer. After the process is running, click Debug / Attach to Process (or press `CTRL+ALT+P`) to open the "Attach to Process" dialog box. You can use this capability to debug apps that are running on a local or remote computer, debug multiple processes simultaneously.

If Electron is running under a different user account, select the `Show processes from all users` check box. Notice that depending on how many BrowserWindows your app opened, you will see multiple processes. A typical one-window app will result in Visual Studio presenting you with two `Electron.exe` entries - one for the main process and one for the renderer process. Since the list only gives you names, there's currently no reliable way of figuring out which is which.

Which Process Should I Attach to?

Code executed within the main process (that is, code found in or eventually run by your main JavaScript file) will run inside the main process, while other code will execute inside its respective renderer process.

You can be attached to multiple programs when you are debugging, but only one program is active in the debugger at any time. You can set the active program in the `Debug Location` toolbar or the `Processes window`.

Using ProcMon to Observe a Process

While Visual Studio is fantastic for inspecting specific code paths, ProcMon's strength is really in observing everything your application is doing with the operating system - it captures File, Registry, Network, Process, and Profiling details of processes. It attempts to log **all** events occurring and can be quite overwhelming, but if you seek to understand what and how your application is doing to the operating system, it can be a valuable resource.

For an introduction to ProcMon's basic and advanced debugging features, go check out this video tutorial provided by Microsoft.

Using WinDbg

It's possible to debug crashes and issues in the Renderer process with WinDbg.

To attach to a debug a process with WinDbg:

1. Add `--renderer-startup-dialog` as a command line flag to Electron.
2. Launch the app you are intending to debug.
3. A dialog box will appear with a pid: "Renderer starting with pid: 1234".
4. Launch WinDbg and choose "File > Attach to process" in the application menu.
5. Enter in pid from the dialog box in Step 3.

6. See that the debugger will be in a paused state, and that there is a command line in the app to enter text into.
7. Type “g” into the above command line to start the debuggee.
8. Press the enter key to continue the program.
9. Go back to the dialog box and press “ok”.