Theseus: Understanding Asynchronous Code

The Problem: asynchronous control flow is convoluted

JavaScript relies on callback functions for events (UI, network, etc.), blocking I/O operations, and control structures like loops. They are error-prone and many libraries exist to help structure them.

Callbacks turn questions about whether a point in the code has been executed into reachability questions that are difficult to answer with ordinary call graphs and most debugging tools.

0 clicks: reachability coloring + call counts

Code that has never been executed is colored gray. Call counts are shown next to every function definition.

Users tend to leave the editor open on one side of the screen and watch code light up as they interact with their application.

1 click: retroactive log

Click a function’s call count to retroactively log the arguments, return value, and any thrown exceptions of every invocation of that function.

Statistics from a lab study about the types of values people print and inspect (without Theseus) suggest that inferring the values to print based on context may be sufficient in most cases:

2 clicks: asynchronous call tree

Invocations with a caller/callee relationship are shown nested in a call tree. Event chains are shown the same way and flagged with ‘async’.

This combination of console and call tree untangles the control flow of complex asynchronous code, visualizing control- and data-flow together.

Download Theseus today!