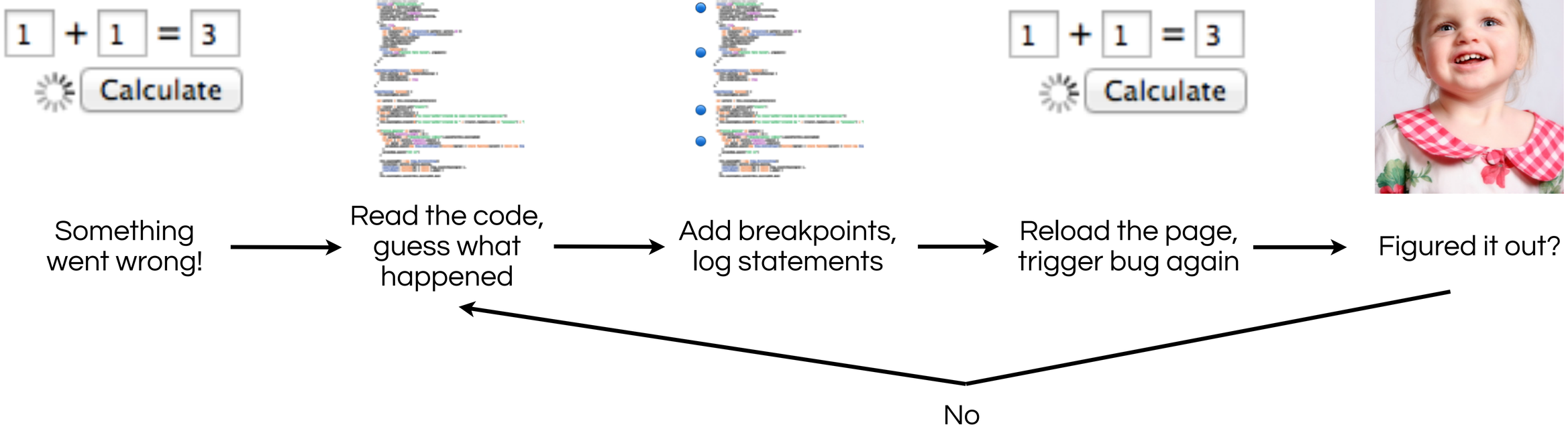
Understanding Asynchronous Code

Interfaces for exposing the run-time behavior of asynchronous JavaScript code on the web

Problem Scenario



Our Insight

Allow developers to test hypotheses while reading the code, like stepping through all traces at once.

```
27 calls    function dispatch(type, e) {
                for (var i in listeners[type]) {
                    listeners[type][i](e);
                }
            }

3 calls     function getExampleData() {
3 calls         $.get("/example.json", { success: function (data) {
                    dispatch("changef", { "data" : data });
                } });
            }

1 call      register("change", function (e) {
                page.render(e.data);
            });
```

See call counts to know what code was hit

```
got here 3 / 3  function dispatch(type, e) {
                    for (var i in listeners[type]) {
                        listeners[type][i](e);
                    }
                }

3 calls         function getExampleData() {
got here 3 / 3      $.get("/example.json", { success: function (data) {
                        dispatch("changef", { "data" : data });
                    } });
                }

got here 0 / 3  register("change", function (e) {
                    page.render(e.data);
                });
```

Click!

- All code reachable from `getExampleData` is highlighted
- Calls to `dispatch` from elsewhere have been filtered from the counts
- Click again to extend the query

Tom Lieber dynamic@mit.edu
Joel Brandt jobrandt@adobe.com
Rob Miller rcm@mit.edu

Adobe

CSAIL
MIT COMPUTER SCIENCE AND ARTIFICIAL INTELLIGENCE LABORATORY