IS2955 - DELIVERABLE 3: Systems Testing with BDD

ThinkThroughMath JavaScript Calculator

Somi Laad and Kayla Mormak

**Summary**

The program being tested is Think Through Math’s Javascript Calculator found at <http://thinkthroughmath.github.io/javascript-calculator/>. This program was chosen for testing because of its use of Javascript, which allowed the use of Gherkin and Cucumber to run BDD tests. Additionally, the testers’ previous experience with this program showed a number of bugs and quirks that could benefit from a narrowed problem definition via testing.

The user stories were chosen to address the needs of a wide scope of users, with special attention to bugs identified during previous testing. In particular, decimal handling has some known bugs regarding decimal and scientific notation in moderately low numbers; the decimal test story is intended to highlight the extent of the problem. Post-error handling issues are addressed by inducing errors and attempting to use them in equations. Finally, minimum complexity is ensured for the use of the calculator in all classrooms.

**Challenges**

The biggest challenge faced by the testers was getting Cucumber to correctly read in the step definition files and apply them to the code under test. Meeting with the professor provided some insights to potential issues, but the most common problems were troubleshot and eliminated. Troubleshooting concluded with the code under test generating an error, which the testers are not able to change or otherwise impact. The window error from the Javascript file is included in the screenshots provided below.

Other challenges with using Cucumber and Node in the future include continued maintenance of installation instructions as operating systems advance. These programs run very easily on a Windows machine, but require drastically more work to implement on Mac OSX. Installation information and troubleshooting for both is limited and largely outdated.

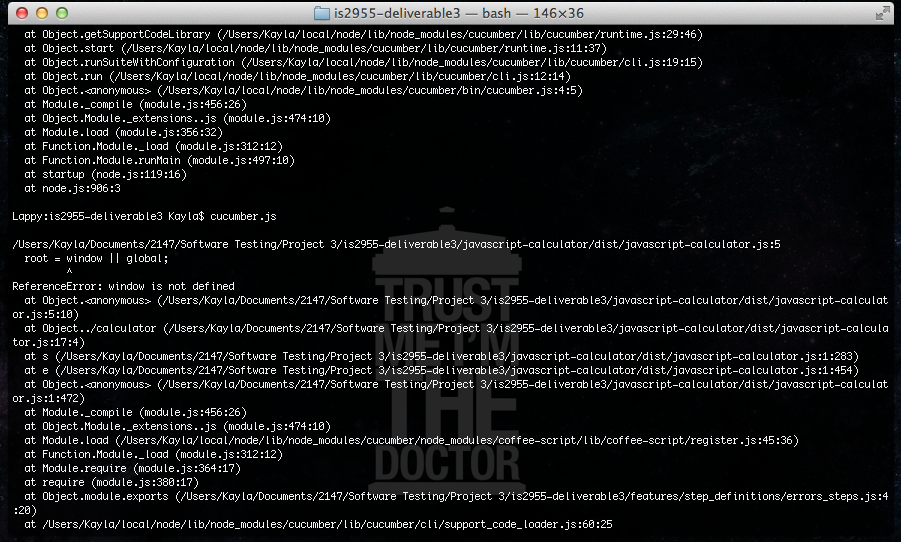
A second challenge for the testers was writing step definitions based on functional Javascript; both have significantly more experience with Object-Oriented programming, which caused a significant amount of confusion when bridging from Gherkin scenarios to Javascript step definitions. Another significant challenge included a difficulty in communication due to one tester’s absence during the testing period. Online communication and source control collaboration were not sufficient enough to address the problems that arose.

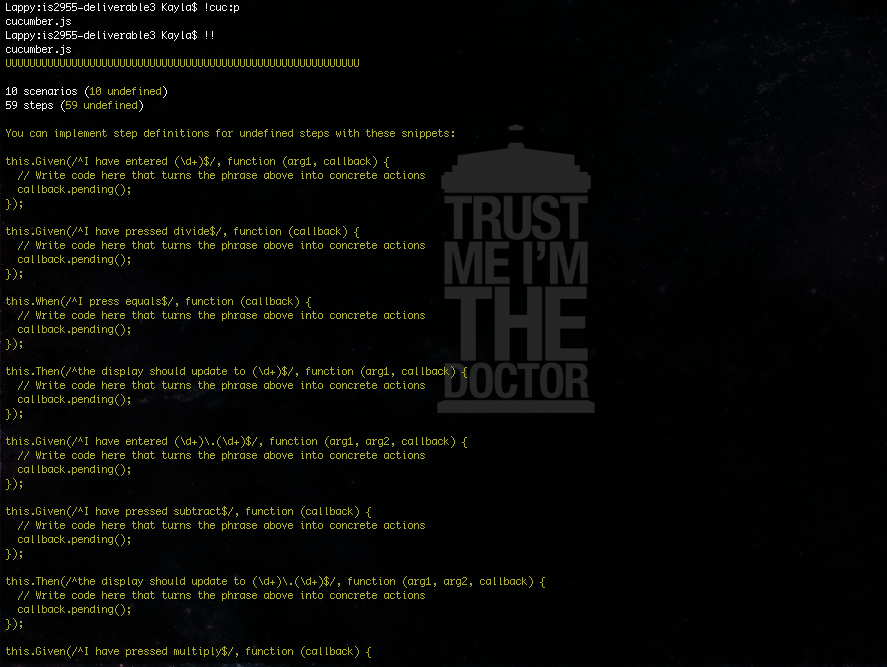
Some tests in this suite were designed to fail, in order to highlight known problems in the calculator functions. For example, 10^-5 and 10^-6 give a result of 0, but 10^-4 shows 0.0001 and 10^-7 shows 1e-7. A scenario was designed to target within these bounds and another was designed to target above them, to show the discrepancy in decimal-number handling.

**Github**

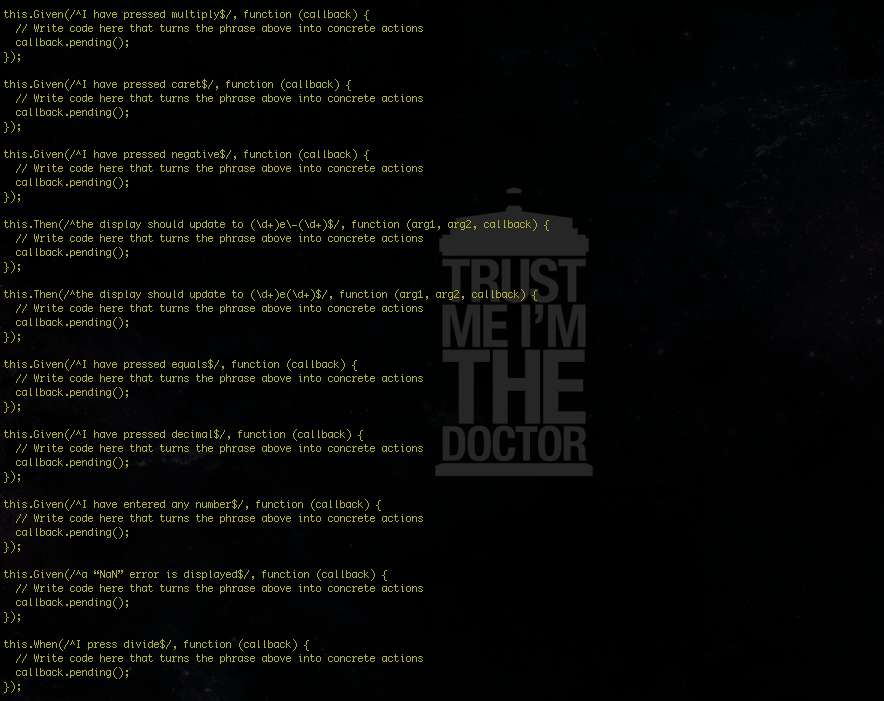
<https://github.com/vinyaa/is2955-deliverable3>

**Code Output**

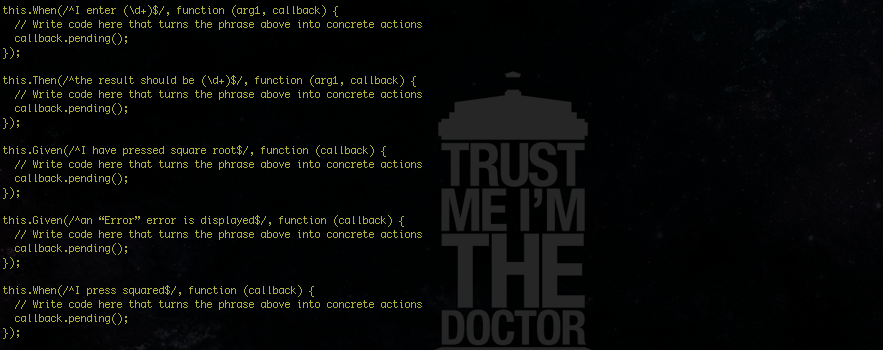
Window Error

All Scenarios, Part 1:

All Scenarios, Part 2



All Scenarios, Part 3



Sample of Single User Story Output

