SC-T-213-VEFF - Web Programming I

Lab Assignment 6 – Server-side Testing and Debugging

Reykjavík University | Deadline: **25th March 2019, 23:59**

The topic of this assignment is: **Testing and debugging server-side JavaScript**.

1 Part 1: Debugging

This first part is to make sure that you know how to connect a debugger to Node.js applications and find relevant information. Note that the task itself is not very useful - it is to simply get every student to start up the debugger at least once. Getting the debugger setup right is also likely the biggest part of this task. If you do not know how to start, consult references [3] in L9 and [2],[3] in L18.

In the supplementary files, you find (modified) solutions to Lab 5. In *mathUser.js*, set breakpoints in lines 3 and 5, then execute the application with a debugger. Write down, in a text file called debug.txt (.txt/plain text, no doc or other file types), the following values:

1. The values of *__filename* and *divResult* (both in local scope) at the breakpoint in line 3
2. The entire call stack (with line/character numbers) at the breakpoint in line 5
   Note: If you use the VSCode debugger, you can right-click on the call stack and choose "Copy Call Stack".

2 Part 2: Unit Testing Math.js

Write unit tests for *math.js* for the following cases:

1. For function *doDivision(a,b)*
   (a) A single test asserting that the function returns the correct division result (to 2 digits after the comma) given two positive numbers.
   (b) Two tests testing two (different) corner or failure cases. Add a comment before each test stating what that case is and why it can be considered a corner or failure case.
2. For function *stringifyDivision(a,b)*
   (a) A single test asserting that the function returns the correct string ("*a* divided by *b* is *result*") given two positive numbers. Use a Stub/Fake (using the Sinon module) to replace the call to the doDivision function.

3 Requirements and Hints

For the overall assignment, the following requirements shall be fulfilled:

1. Tests shall not contain random numbers. Instead, each test shall test specific/deterministic parameters (see slides 18 in L18).

2. Apart from built-in Node.js modules (and of course the *math.js* module), only *mocha*, *chai*, and *sinon* are permitted. The dependencies are already set up in the *package.json* file in the supplementary material.

Submission

The lab is submitted via Canvas. The following deliverables need to be included:

1. A single zip file containing the file *debug.txt* for Part 1 and the file *math.test.js* for Part 2. Do not include the *node_modules* folder and/or the *package-lock.json*.