CS320 - Homework #5

Due Dec 9th, 2020

Automated Testing with Javascript

1. Introduction

In this assignment, you will practice writing test suites for the primeFunctions code you developed in homework 4, using Mocha (https://mochajs.org/) and Chai (https://www.chaijs.com). Note that by default, you should have ESLint in place for this as was needed for Homework 4. Also, you should use GitHub for version control as directed.

2. Test Suite 1 – Correctness

This test suite is for testing the correctness of the three functions you wrote in homework 4, primeGen, cumulativeSum, and maxPrimeSum.

1) **primeGen:** Write two test cases:

```
primeGen(10)=>[2, 3, 5, 7]
primeGen(20)=>[2, 3, 5, 7, 11, 13, 17, 19]
```

2) **cumulativeSum:** Write two test cases:

```
cumulativeSum([1, 2, 3, 4])=>[1, 3, 6, 10]
cumulativeSum([10, 11, 12, 13, 14])=>[10, 21, 33, 46, 60]
```

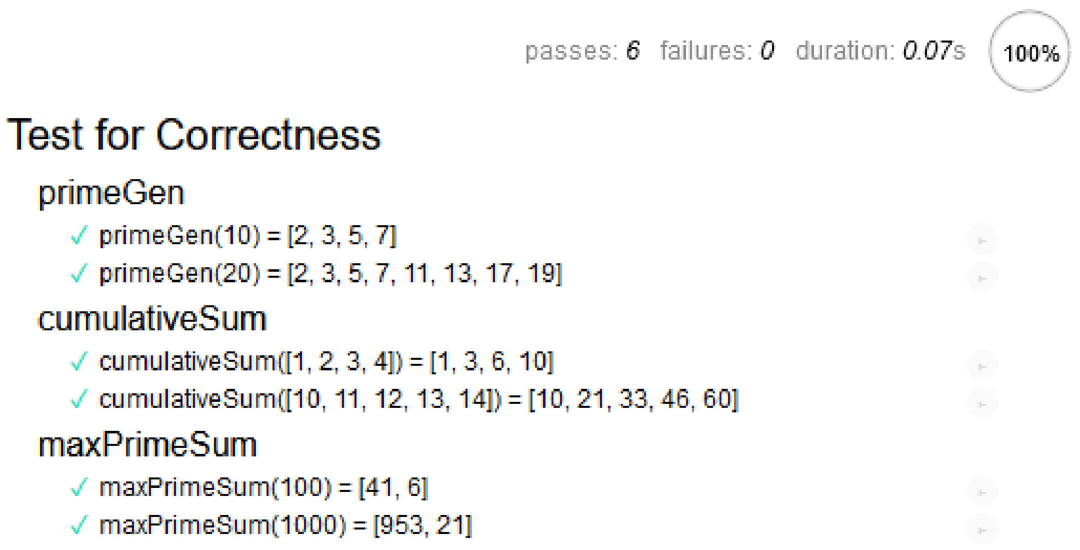
3) **maxPrimeSum:** Write two test cases:

```
maxPrimeSum(100)=>[41, 6]
maxPrimeSum(1000)=>[953, 21]
```

If you run this test suite in a browser, it should look like the following picture:

CS320 - Homework #5

Due Dec 9th, 2020



```
passes: 6   failures: 0   duration: 0.07s   100%

Test for Correctness
  primeGen
    ✓ primeGen(10) = [2, 3, 5, 7]
    ✓ primeGen(20) = [2, 3, 5, 7, 11, 13, 17, 19]
  cumulativeSum
    ✓ cumulativeSum([1, 2, 3, 4]) = [1, 3, 6, 10]
    ✓ cumulativeSum([10, 11, 12, 13, 14]) = [10, 21, 33, 46, 60]
  maxPrimeSum
    ✓ maxPrimeSum(100) = [41, 6]
    ✓ maxPrimeSum(1000) = [953, 21]
```

3. Test Suite 2 – Performance

Optimizing the code that has already been tested is often needed. It is always good to know the execution time of a test case. Mocha uses a threshold called slow to flag the test cases that take too long to run. The default value for this parameter is 75ms. You can customize this threshold in your test cases using this.slow(*yourthreshold*).

For example, if you add the following statement to one of your test cases:
this.slow(100)
when you run the test, Mocha will generate a red flag if the execution time of this test is longer than 100ms. This is a useful tool which helps you to understand the performance of your tests.

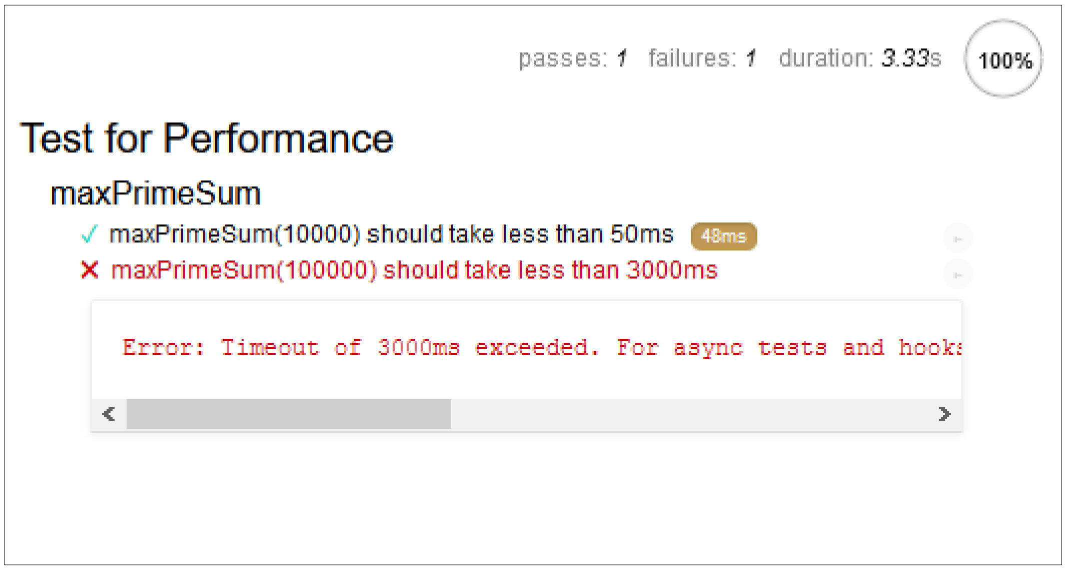
Note that using the parameter slow won't make the test case fail, even if it takes longer than the specified threshold. If you would like to make the test case fail, you can use this.timeout.

For example, if you add the following statement to one of your test cases:
this.timeout(1000)
when you run the test, it will fail if it takes longer than 1000ms. This is how you test the performance of your code.

In the Test Suite 2, you will create a test suite to test the performance of your maxPrimeSum function. Specifically, write two test cases:

```
maxPrimeSum(10000)=>[9521, 65]
maxPrimeSum(100000)=>[92951, 183]
```

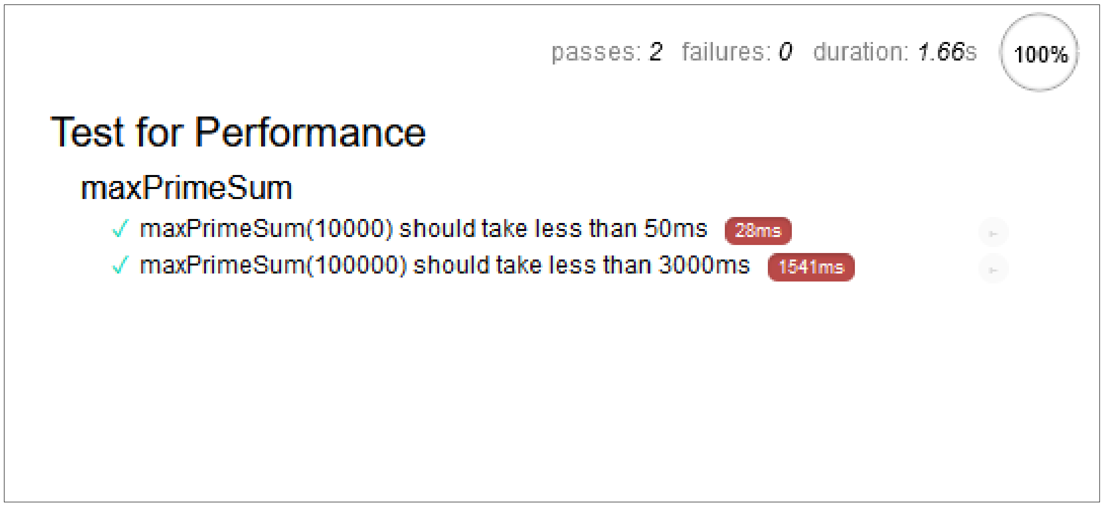
Set the slow threshold to be 0, so you always get the performance data when you run your test. Based on your observation of the performance, you should set up a goal for your optimization, for example: 50ms and 3000ms for those two functions. Use this.timeout to make the tests fail if they don't satisfy these performance requirements as shown below:



```
passes: 1   failures: 1   duration: 3.33s   100%

Test for Performance
  maxPrimeSum
    ✓ maxPrimeSum(10000) should take less than 50ms  48ms
    ✗ maxPrimeSum(100000) should take less than 3000ms
      Error: Timeout of 3000ms exceeded. For async tests and hook
```

Test of Pre-optimized Code



```
passes: 2   failures: 0   duration: 1.66s   100%

Test for Performance
  maxPrimeSum
    ✓ maxPrimeSum(10000) should take less than 50ms  28ms
    ✓ maxPrimeSum(100000) should take less than 3000ms  1541ms
```

Test of Optimized Code

Then you should optimize your code to have an improved performance and document the optimizations. After that, run the correctness testing (Test Suite 1) to make sure that your optimization didn't break the correctness of the code. This is what we call *regression testing*.

CS320 - Homework #5
Due Dec 9th, 2020

4. Workflow Instructions

You must follow the step-by-step workflow instructions below:

1. You will start from the "primefunctions" repo you created in Assignment 4.

2. Create a branch, call it "testsuite1", and start the work of Homework 5 on this branch.

3. Create a js file for the testing code, call it "testCorrectness.js".

4. Create an html file for displaying the test results, call it "testSuite1.html".

5. Start to work on Test Suite 1, as described in Section 2. Make commits to your repo as needed. Example commits: "finished primeGen tests", "finished cumulativeSum tests", "finished max-PrimeSum tests", etc. (Note that these are just examples, you can make your commits using the labels you find useful).

6. Once you are done with Test Suite 1, merge it back to the master branch.

7. Create a new branch, call it "testsuite2", and start working on the performance tests using this branch.

8. Create a copy of your primefunctions.js file, name it "primefunctions2.js", you can do your optimization in this copy.

9. Create a js file for the testing code, call it "testPerformance.js".

10. Create an html file for displaying the test results, call it "testSuite2.html".

CS320 - Homework #5

Due Dec 9th, 2020

11. Start to work on Test Suite 2, as described in Section 3. Make commits to your repo as needed. Example commits: "finished tests", "optimization 1", "optimization 2", "finished optimization", etc. (Note that these are just examples, you can make your commits using the labels you find useful). Record what your different optimizations were in a comment block at the bottom of your primefunctions2.js file.

12. Once you are done with Test Suite 2, merge it back to the master branch.

13.Make sure that there is a green check mark indicating that ESLint does not detect any problems in your code.

14. Do some final cleanup if necessary. Commit your project to GitHub, label it "Finished Assignment 5." Check that you code is on GitHub. Keep a screenshot of all your commits on GitHub. Recall, Homework GitHub entries should be private until after the semester.

5. Submission

This assignment is due at 11:59pm on 12/9/2020. Late submissions are subject to a 20% penalty. Late submissions will not be accepted after 3 days past the deadline, in this case 11:59pm on 12/12/2020.Submit the following files in a package (.zip or .tar) to Blackboard:

1. primefunctions.js (original version)

2. primefunctions2.js (optimized version)

3. testCorrectness.js

4. testSuite1.html

5. testPerformance.js

6. testSuite2.html

7. Screenshots of test suite 1, slower test suite 2, and faster test suite 2, in either as png or PDF files.

8. GitHub screenshot showing all your commits, in either png or PDF file.

6. Grading

This assignment will be graded out of 50. For your information, the grading scheme is shown in the following table.

20pts	Test Suite 1
15pts	Test Suite 2
10pts	Optimizations
5pts	Use of Git
10pts	Code quality (no ESLint errors)

TOTAL: 60 pts