COMP.SE.200-2020-2021-1 Software Testing
Test report

Jussi Kujanen 273161
Veikko Hiltunen 283475
https://github.com/Kujanenj/SoftwareTesting

Tampere University
October 10th, 2020

Contents

1 Introduction

The purpose of this document is to describe testing results for an application which is used to sell food products. Testing was done only for the most important functions of the program due to time limitations. The aim of testing was to find problems that easily break the program. All tests cases are listed in test cases section and overall results are covered in findings and conclusions section.

2 Test cases

This chapter will cover unit and integration testing. With the large amount of functions present, prioritisation is a must. The functions are rated on a scale of 1 to 5, 5 having the highest priority. Based on the description of the use case for the library, we can focus mostly on functions that handle string manipulation and validation, such as 'capitalize','map' and 'isEmpty'. This is because the part of the frontend to be tested is mostly based on handling user input.

Functions that are not essential for string manipulation have lower priority, and are not tested. A complete list of module prioreties and reasoning can be found as an attachment to this document. 1

2.1 Unit testing and integration testing

Unit tests and integration tests were run using Jest. We also used TravisCI to make all commits automatically tested and Coveralls which gave percentage of how many lines were properly covered.

The functions to be tested were selected according to their importance to the program. Almost all functions that were in no way related to the operation of the program have been excluded from testing.

2.1.1 Test cases

Here will be listed all unit and integration tests and their results. All original tests are underlined.

1. Function: add.js

Tests:

- Unit - add
  - Only positive numbers
  - Negative and positive numbers
  - Only negative numbers
  - Zeros
- Integration
  - Strings using toString

Result: All tests passed without problems.

2. Function: capitalize.js

Tests:

- Unit - correct capitalization when
  - All characters are capitalized
  - First letter is number
  - First letter is character and others are numbers
  - String contains other symbols

Result: All tests passed without problems. We decided to add more functions because we wanted to make sure that the function works with other values aswell.

3. Function: compact.js

Tests:

- Unit - compact
  - Large item id list

Result: When we gave array to compact.js function it doesn't filter values properly. For example, when we gave array [0, 1, false, 2, '', 3] it returned [0, 1, false, 2, ,3] which is far from right one which is [1, 2, 3]. That's why we decided to not test that function anymore because it broke our pipeline.

4. Function: Endswith.js

Tests:

- Unit - when string endswith
  - Valid character
  - Invalid character
  - Second to last character

Result: All tests passed without problems.

5. Function: filter.js

Tests:

- Unit - filter
  - Invalid product numbers
  - Sold products
- Integration
  - Ending with filter

Result: All tests passed without problems.

6. Function: get.js

Tests:

- Unit - get object
  - When there is no field
  - When there is valid test object
  - When there is valid test object but the path is wrong

Result: All tests passed without problems.

7. Function: isEmpty.js

Tests:

- Unit - is empty
  - When value is null
  - When value is array
  - When value is object
  - When value is number

Result: All tests passed without problems. Here, we also wanted to add more tests so we can be sure that function can work between different parameters.

8. Function: map.js

Tests:

- Unit - map
  - To capitalize multiple products
  - To increase multiple product prices
  - To capitalize object names
- Integration
  - Falsy list compact

Result: All unit tests passed but we could not test integration because compact function does not work properly.

9. Function: reduce.js

Tests:

- Unit - reduce
  - Sum array
- Integration
  - Sum bad array with compact

Result: All unit tests passed without problems but integration test failed because of compact function.

10. Function: slice.js

Tests:

- Unit - slice
  - End of the array
  - From between of the array

Result: All unit tests passed without problems.

11. Function: toNumber.js

Tests:

- Unit - to number
  - when value is string
  - when value is infinity
  - when value is decimal
  - when value is negative integer

Result: All unit tests passed without problems.

12. Function: toString.js

Tests:

- Unit - to string
  - when value is array
  - when value is negative zero
- Integration
  - String: slice of objects

Result: All unit tests passed without problems.

3 Findings and conclusions

Overall unit tests went pretty well when integration testing was a bit rough because of failure in compact.js function. We noticed that it was easy to test more deeply all functions and that is very noticeable in the number of tests compared to the original plan. Also TravisCI and Coveralls together helped a lot because it was so much easier to write new tests when you knew what parts of the function were not tested yet. In addition, it was great to see how easy it was to setup pipeline which told after certain amount of time was the build succesful and whether the results improved in any way. In the future, we are sure that we will continue to use Travis and Coveralls to get all data out of our code.

References

Attacments

Priority table 1

Table 1: Priority

Module Name	**Importance Rating**	**Reason**
add	4	Adding prices
at	1	Not needed
camelCase	1	Not needed
capitalize	3	String manipulation
castArray	1	Not needed
ceil	1	Not needed
chunk	1	Not needed
clamp	1	Not needed
compact	2	Might need to remove empty values
countBy	1	Not needed
defaultTo	1	Not needed
defaultToAny	1	Not needed
difference	1	Not needed
divide	1	Not needed
drop	1	Not needed
endsWith	3	Searching for products
eq	1	Not needed
every	1	Not needed
filter	3	Searching products
get	2	object manipulation
isArguments	1	Not needed
isArrayLike	1	Not needed
isArrayLike	1	Not needed
isBoolean	1	Not needed
isBuffer	1	Not needed
isDate	1	Not needed
isEmpty	2	Empty input fields
isLenght	1	Not needed
isObject	1	Not needed
isObjectLike	1	Not needed
isSymbol	1	Not needed
isTypedArray	1	Not needed
keys	1	Not needed
map	3	String manipulation
memoize	1	Not needed
reduce	4	String manipulation, adding prices
slice	2	Remove items from cart
toFinite	1	Not needed
toInteger	2	Input validation
toNumber	2	Input validation
toString	2	Input validation, string manipulation
upperFirst	1	Not needed
words	1	Not needed