Test report

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

Jussi Kujanen 273161
Veikko Hiltunen 283475

Github

Tampere University
Finland
November 17th, 2020

Contents

1 Introduction

The purpose of this document is to report 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, and tests added in the second phase of 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 [1]. We also used TravisCI [2] to make all commits automatically tested and Coveralls [3] 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.

Functions that were not part of the original test plan, have their test number **emboldened**. Tests that are integration tests are marked with [i].

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

Result: All tests passed without problems.

Num	Test case	Test data	Expected result	Actual result	Passed
1	positive numbers	(0, 5)	5	7	Yes
2	negative and positive numbers	(-5, 6)	1	1	Yes
3	only negative numbers	(-5, 6)	-11	-11	Yes
4	zeros	(0, 0)	0	0	Yes

2. Function: capitalize.js

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.

Num	Test case	Test data	Expected result	Actual result	Passed
1	when all characters are capitalized	(TEST STRING)	Teststring	Teststring	Yes
2	when first letter is number	(0123Test)	0123test	0123test	Yes
3	when first letter is character and others are numbers	(P1234567)	P1234567	P1234567	Yes
4	when string contains other symbols	(J%/¤)	J%/¤	J%/¤	Yes

3. Function: compact.js

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.

Num	Test case	Test data	Expected result	Actual result	Passed
1	large item_id list	[0, 1, false, 2, ", 3]	[1, 2, 3]	[0, 1, false, 2, ,3]	no

4. Function: Endswith.js

Result: All tests passed without problems.

Num	Test case	Test data	Expected result	Actual result	Passed
1	valid character	"abc","c"	true	true	Yes
2	invalid character	abc,b	false	false	Yes
3	second to last character	abc,b, 2	true	true	Yes

5. Function: filter.js

Result: All tests passed without problems.

NOTE:

products =
'product' : 'dogfood', 'unsold' : true
'product' : 'catfood', 'unsold' : false

Num	Test case	Test data	Expected result	Actual result	Passed
1	invalid product numbers	[1, 2, 3, 4, 5, -6, 7]	[1, 2, 3, 4, 5, 7]	[1, 2, 3, 4, 5, 7]	Yes
2	sold products	products	dogfood	dogfood	Yes
3[i]	ending with filter	["ab", "ac", "bb"], b	["ab", "bb"]	["ab", "bb"]	Yes

6. Function: get.js

Result: All tests passed without problems.

NOTE: testobject = { a : { b : 1} }

Num	Test case	Test data	Expected result	Actual result	Passed
1	Non existent field	{a},'a.b'	undefined	undefined	Yes
2	Valid object	testObject, 'a.b'	1	1	Yes
3	Not equal 2	testObject, 'a.b'	not 2	not 2	Yes

7. Function: isEmpty.js

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.

Num	Test case	Test data	Expected result	Actual result	Passed
1	Value is null	null	True	true	Yes
2	Value is array	[1,2,3]	false	false	Yes
3	Value is object	{a:1}	false	false	Yes
4	Value is number	1	True	true	Yes

8. Function: map.js

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

products = [{ 'name': 'chair', 'price': 20 }, { 'name': 'kalle', 'price': 50 }]

Num	Test case	Test data	Expected result	Actual result	Passed
1	Capitalize multiple products	[product1, product2], capitalize	[Product1, Product2]	[Product1, Product2]	Yes
2	Increase multiple product prices	products, price * 2	[40,100]	[40,100	Yes
3	Capitalize object names	products, capitalize	[Chair, Kalle]	[Chair, Kalle]	Yes
4[i]	Falsy list compact	compact, map [1,2,3,4] isEmpty	[2,3,4,]	-	No

9. Function: reduce.js

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

Num	Test case	Test data	Expected result	Actual result	Passed
1	Sum of an array	[1,2,3],add,0	6	6	Yes
2[i]	Sum bad array	compact([1,2,3,""], add,0)	5	-	No

10. Function: slice.js
All unit tests passed without problems.

Num	Test case	Test data	Expected result	Actual result	Passed
1	Slice start	[1, 2, 3, 4] , 2	[3, 4]	[3, 4]	Yes
2	Middle of the array	[1, 2, 3, 4, 5, 6] , 1,5	[2.3.4.5]	[2.3.4.5]	Yes

11. Function: toNumber.js

Result: All unit tests passed without problems.

Num	Test case	Test data	Expected result	Actual result	Passed
1	modify string to number	'3.2'	3.2	3.2	Yes
2	"Value is infinity	infinity	infinity	infinity	Yes
3	Value is decimal	500.2	500.2	500.2	Yes
4	Value is negative integer	-73	-73	-73	Yes

12. Function: toString.js

Result: All unit rests passed without problems.

Num	Test case	Test data	Expected result	Actual result	Passed
1	multiple products into a single string	[p1, p2, p3, p4]	'p1, p2, p3, p4'	'p1, p2, p3, p4'	Yes
2	When value is negative zero	-0	"-0"	"-0"	Yes
3	String: Slice of objects	toString(get(slice(['a':1 , 'b':2], 0, 1)[0], 'a')))	"1"	"1"	Yes

3 Findings and conclusions

Most of the unit- and integration tests were passed without any problems. The only bug found was with compact.js. The module functionality violates function description, and thus usage should be avoided untill a patch is provided.

Some test cases were skipped, such as "Falsy list compact" in map.js. These test cases were skipped because of the bug in compact.js mentioned earlier.

The biggest problem the tested library has, in our opinion, was the inconsistency of edge case descriptions. Some functions would report results as undefined or null, while some would result in a crash due to internal errors. Extra care would have to be taken when using this library during edge case usage.

In the end, a Coveralls reported a 73% test coverage for the library. We consider this a sufficient coverage, considering that a significant portion of the library was left untested, due to the reasons described in the previous document.

That being said, nothing can be fully tested, and a complete reliance that a software is bug free should be avoided. Considering this, new tests are always welcome.

References

[1] Jest, [internet] https://jestjs.io/, [17.11.2020].

[2] Travis, [internet] https://travis-ci.org/, [17.11.2020].

[3] Coveralls, [internet] https://coveralls.io/, [17.11.2020].

Attacments

Priority table 1
Link to github document: https://github.com/Kujanenj/SoftwareTesting

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