

HEADING

Jussi Kujanen 273161 Veikko Hiltunen 283475

Tampere University
Päivät tänne

Contents

1	Definitions	1
2	Introduction	1
3	Tools	1
3.1	Jest	1
3.2	TravisCI	1
3.3	Coveralls	2
4	Test cases	2
4.1	Unit testing	2
4.1.1	Test cases	2
4.2	Integration testing	5
4.2.1	Test cases	5
5	References	5

1 Definitions

Definition 1. *This is a definition*

2 Introduction

The purpose of this document is to describe testing for an application which is used to sell food products. Testing is done only for the most important functions of the program due to time limitations. The aim of testing is to find problems that easily break the program. In addition, the purpose of this document is to help the reader understand how the testing and the functions to be tested have been roughly performed.

3 Tools

Below this section is a description of all the tools that will be used for this project. The tools have been chosen according to our own experience and interest.

3.1 Jest

Jest[\[1\]](#) was chosen as the testframework, since it works well with React, and we both have had some experience with Jest beforehand. Jest is used to run the test cases described in this document. A tutorial explaining how to use jest can be found here.[\[2\]](#)

3.2 TravisCI

TravisCI[\[3\]](#) is used during integration testing, because its automatic building and code testing during changes. Its automation provides immediate feedback which is very useful even in small projects like this. It is also useful in continuous integration, since it is better to merge small code changes frequently than a large commit at the end of the development. Because CI detects deficiencies at the early stage in development, defects are typically less complex and smaller which makes them easier to resolve. Documentation and tutorials for Travis CI usage can be found on this site. [\[4\]](#)

3.3 Coveralls

Coveralls[5] is used to track test coverage. Coveralls was chosen for this task, since it is required by the course, and it is open source. A tutorial on how to use Coveralls can be found here.[6].

4 Test cases

With the large amount of functions present, prioritisation is a must. 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 list¹ of module priorities and reasoning can be found as an attachment to this document.

4.1 Unit testing

Unit tests are run using jest. For each function tested, a table is made, explaining the tests run on that function, their inputs, outputs and expected results.

Each test is considered 'passed' once the output matches the expected output. If bugs or errors are encountered, the corresponding table has a note of it. The complete coverage table can be found in the attachments² at the bottom of this document.

4.1.1 Test cases

Function: add.js

Num	Test case	Test data	Expected result	Actual result	Passed
1	Positive numbers	0 + 5	5	-	No
2	Negative numbers	-5 + 6	1	-	No

Function: capitalize.js

Function: compact.js

Function: filter.js

Function: map.js

Function: slice.js

Num	Test case	Test data	Expected result	Actual result	Passed
1	all characters capitalized	TESTSTRING	Teststring	-	No

Num	Test case	Test data	Expected result	Actual result	Passed
1	large item_id list	[0, 1, ", 5, 6]	[0,1,5,6]	-	No

Num	Test case	Test data	Expected result	Actual result	Passed
1	filter products	[1,2,3,4,5,-5,-3]	[1,2,3,4,5]	-	No

Num	Test case	Test data	Expected result
1	similar actions for array nodes	[product1, product2], capitalize	[Product1, Product2]

Num	Test case	Test data	Expected result	Actual result
1	remove certain amount from product list	[1, 2, 3, 4] , 2	[3, 4]	-

Function: toNumber.js

Num	Test case	Test data	Expected result	Actual result	Passed
1	modify string to number	'3,2'	3,2	-	No

Function: toString.js

Num	Test case	Test data	Expected result	Actual result
1	multiple products into a single string	[p1, p2, p3, p4]	'p1, p2, p3, p4'	-

Function: endWith.js

Num	Test case	Test data	Expected result	Actual result	Passed
1	Valid input	"abc","b"	False	-	No
2	Invalid input	abc,b	???	-	No

Function: get.js

Num	Test case	Test data	Expected result	Actual result	Passed
1	Non existent field	{a},'a.b'	???	-	No
2	valid Input	{a:b:1},'a.b'	???	-	No

Function: isEmpty.js

Num	Test case	Test data	Expected result	Actual result	Passed
1	Null	null	False	-	No

Function: reduce.js

Num	Test case	Test data	Expected result	Actual result	Passed
1	Sum of an array	[1,2,3],add,0	6	-	No

4.2 Integration testing

4.2.1 Test cases

5 References

References

- [1] [Online]. Available: <https://jestjs.io/>
- [2] [Online]. Available: <https://www.valentinog.com/blog/jest/>
- [3] [Online]. Available: <https://travis-ci.org/>
- [4] [Online]. Available: <https://docs.travis-ci.com/user/tutorial/>
- [5] [Online]. Available: <https://coveralls.io/>
- [6] [Online]. Available: <https://github.com/dwyl/learn-coveralls.io>

Attacments

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

Table 2: Test cases

Target file	Test name	Purpose of test	Test data	Expected result	Actual result
1. add.js	limit values	Product price uses this function	0 + 5	5	-
2. capitalize.js	all characters capitalized	Product descriptions must start with an upper case letter	TESTSTRING	Teststring	-
3. compact.js	large item_id list	Tests large item_id list and deletes unnecessary values	[0, 1, ", 5, 6]	[0,1,5,6]	-
4. filter.js	filter products	Filter invalid product id's	[1,2,3,4,5,-5,-3]	[1,2,3,4,5]	-
5. map.js	similar actions for array nodes	Multiple products need to make same actions	[product1, product2], capitalize	[Product1, Product2]	-
6. slice.js	remove certain amount from product list	Easier to delete multiple products	[1, 2, 3, 4] , 2	[3, 4]	-
7. toNumber.js	modify string to number	large string list to numbers	'3,2'	3,2	-
8. toString.js	multiple products into a single string	Seperate products from list to a single string	[p1, p2, p3, p4]	'p1, p2, p3, p4'	-
9. endWith.js	Valid input	String validation	"abc","b"	False	-
10. endWith.js	Invalid input	String validation - check for crash	abc,b	????	-
11. get.js	Non existent field	Check for crash	{a},'a.b'	???	-
12. get.js	valid Input	For several keywords, might need to check every word	{a:b:1},'a.b'	1	-
13. isEmpty.js	Null	Validate empty inputs	null	False	-
14. reduce.js	Sum of an array	Need to add ⁷ prices of items in cart	[1,2,3],add,0	6	-