**Technical specifications for “Easy - Simple addition”**

Linus Lindgren

March 25.

sfx-o-assign2-persisted-addition

Clone this repository Create a branch

New requirements that need to be met:

The user inserted numbers need to be persisted

The function that handles read/writing the persisted numbers should also contain the summarizing logic

The function above should be unit tested

Commit Push your branch Create a pull request =>

Done!

**Explanatory note to the technical specifications for “Easy - Simple addition”**

**What needs to be done?**

1. Move the logic of saving, receiving and summing numbers into a separate function.

2. Write unit tests for this function.

3. Make sure everything works correctly.

**1. Updated code with modular function for working with numbers**

**1.1 Project structure**

my-calculator-app/

│── src/ stores all the source code

│ ├── utils/

│ │ ├── calculator.js calculation functions (e.g. sum, average, etc.) (no DOM)

│ │ ├── storage.js logic for saving numbers to localStorage

│── tests/ all tests

│ ├── calculator.test.js tests for mathematical functions

│ ├── storage.test.js tests for saving and loading numbers

│──app.js UI handling (button clicks, interface updates)

│──index.html the calculator interface itself

│── main.js browser code here

│── package.json stores dependencies (jest, eslint, babel, etc.)

│── jest.config.mjs Jest configuration

│──.gitignore excludes unnecessary files (node\_modules/, dist/)

**1.2. Let's create a separate file storage.js, which will be responsible for saving and processing numbers:**

const STORAGE\_KEY = 'numbers';

export function getStoredNumbers() {

return JSON.parse(localStorage.getItem(STORAGE\_KEY)) || [];

}

export function saveNumbers(numbers) {

localStorage.setItem(STORAGE\_KEY, JSON.stringify(numbers));

}

export function addNumberToStorage(number) {

const numbers = getStoredNumbers();

numbers.push(number);

saveNumbers(numbers);

return numbers; // Возвращаем новый список чисел

}

export function calculateSum(numbers) {

return numbers.reduce((sum, num) => sum + num, 0);

}

//with the addition function:

export function sum(a, b) {

return a + b;

}

**1.3. main.js (main UI logic)**

import { getStoredNumbers, addNumberToStorage, calculateSum } from './utils/storage.js';

document.addEventListener('DOMContentLoaded', () => {

const numberInput = document.getElementById('numberInput');

const addButton = document.getElementById('addButton');

const numberList = document.getElementById('numberList');

const totalSum = document.getElementById('totalSum');

function updateUI(numbers) {

numberList.textContent = numbers.join(' + ') || "No numbers yet";

totalSum.textContent = calculateSum(numbers);

}

function addNumber() {

const number = numberInput.value.trim();

if (!/^-?\d+$/.test(number)) {

alert('Please enter a valid integer.');

return;

}

const numValue = parseInt(number, 10);

const updatedNumbers = addNumberToStorage(numValue);

updateUI(updatedNumbers);

numberInput.value = '';

numberInput.focus();

}

addButton.addEventListener('click', addNumber);

// Восстанавливаем сохраненные числа при загрузке

updateUI(getStoredNumbers());

});

**2. Unit tests (storage.test.mjs)**

**2.1. Now let's write tests for storage.js. We'll use Jest for this.**

import { getStoredNumbers, addNumberToStorage, calculateSum } from '../src/utils/storage.js';

describe('Storage and Summation Tests', () => {

beforeEach(() => {

localStorage.clear();

});

test('should return empty array if no numbers are stored', () => {

expect(getStoredNumbers()).toEqual([]);

});

test('should add number to storage and return updated array', () => {

expect(addNumberToStorage(5)).toEqual([5]);

expect(addNumberToStorage(10)).toEqual([5, 10]);

});

test('should correctly sum stored numbers', () => {

addNumberToStorage(3);

addNumberToStorage(7);

addNumberToStorage(-2);

expect(calculateSum(getStoredNumbers())).toBe(8);

});

});

//with the addition function:

import { sum } from '../src/utils/storage.js';

test('adds 1 + 2 to equal 3', () => {

expect(sum(1, 2)).toBe(3);

});

**2.2. Testing the program**

### **Let's run the basic test from calculator.test.js:**

import { add } from "../src/utils/calculator.js"; // проверяем, что путь верный

test("adds 2 + 3 to equal 5", () => {

expect(add(2, 3)).toBe(5);

});

### ****2.2. Test result****

\my-calculator-app> npm test

> my-calculator-app@1.0.0 test

> jest

PASS tests/calculator.test.js

√ adds 2 + 3 to equal 5 (5 ms)

Test Suites: 1 passed, 1 total

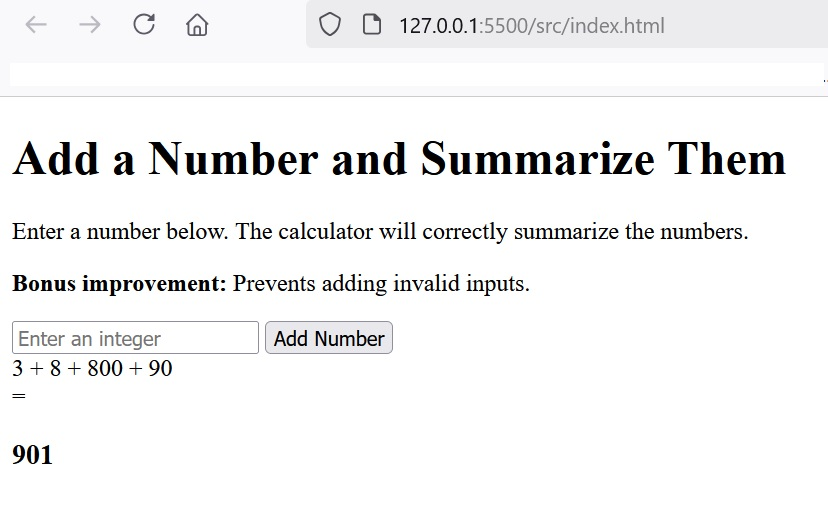
Tests: 1 passed, 1 total

Snapshots: 0 total

Time: 2.085 s, estimated 5 s

Ran all test suites.

**2.3. Testing the project in a browser via Live Server**



**3. Conclusion**

The logic of saving, receiving and summing numbers is moved to a separate function.

Unit tests for this function are written.

Everything works correctly.