# Lab: Unit Testing with JavaScript

Problems for exercises and homework for the ["Back-End Technologies Basics"]() course @ SoftUni.
You can check your solutions in [Judge]().

You are required to **submit only the unit tests** for the **object / function** you are testing.

## 1. Sum of Numbers

Write tests to check the functionality of the following code:

**sumNumbers.js**

```
function sum(arr) {
    let sum = 0;
    for (let num of arr){
        sum += Number(num);
    }
    return sum;
}
```

Your tests will be supplied with a function named **'`sum()`'**. It should meet the following requirements:

- Take an **array of numbers** as an argument
- **Return** the **sum** of the values of **all elements** inside the array

## 2. Check for Symmetry

Write tests to check the functionality of the following code:

**checkForSymmetry.js**

```
function isSymmetric(arr) {
    if (!Array.isArray(arr)){
        return false; // Non-arrays are non-symmetric
    }
    let reversed = arr.slice(0).reverse(); // Clone and reverse
    let equal = (JSON.stringify(arr) == JSON.stringify(reversed));
    return equal;
}
```

Your tests will be supplied with a function named **'`isSymmetric()`'**. It should meet the following requirements:

- Take an **array** as an argument
- **Return false** for any input that isn't of the **correct type**
- **Return true** if the input array is **symmetric**
- Otherwise, **return false**

## 3. RGB to Hex

Write tests to check the functionality of the following code:

**rgb-to-hex.js**

```
function rgbToHexColor(red, green, blue) {
    if (!Number.isInteger(red) || (red < 0) || (red > 255)){
        return undefined; // Red value is invalid
    }
    if (!Number.isInteger(green) || (green < 0) || (green > 255)){
        return undefined; // Green value is invalid
    }
    if (!Number.isInteger(blue) || (blue < 0) || (blue > 255)){
        return undefined; // Blue value is invalid
    }
    return "#" +
        ("0" + red.toString(16).toUpperCase()).slice(-2) +
        ("0" + green.toString(16).toUpperCase()).slice(-2) +
        ("0" + blue.toString(16).toUpperCase()).slice(-2);
}
```

Your tests will be supplied with a function named '**`rgbToHexColor()`**', which takes **three arguments**. It should meet the following requirements:

- Take three **integer numbers**, representing the red, green, and blue values of RGB color, each **within the range [0…255]**
- **Return** the same color in hexadecimal format as a **string** (e.g. **'#FF9EAA'**)
- **Return undefined** if **any** of the input parameters are of an **invalid type** or **not** in the **expected range**

## 4. Add / Subtract

Write tests to check the functionality of the following code:

**addSubtract.js**

```
function createCalculator() {
    let value = 0;
    return {
        add: function(num) { value += Number(num); },
        subtract: function(num) { value -= Number(num); },
        get: function() { return value; }
    }
}
```

Your tests will be supplied with a function named '**`createCalculator()`**'. It should meet the following requirements:

- **Return a module** (**object**), containing the functions **`add()`**, **`subtract()`** and **`get()`** as **`properties`**
- Keep an **internal sum** that **can't be modified** from the outside
- The functions **`add()`** and **`subtract()`** take a parameter that can be **parsed as a number** (either a number or a string containing a number) that is added or subtracted from the **internal sum**
- The function **`get()`** **returns** the value of the **internal sum**