# JS Advanced - Exam

## Problem 3. Unit Testing

### Your Task

Using **Mocha** and **Chai** write **JS Unit Tests** to test a variable named **stringOperations**, which represents an object. You may use the following code as a template:

|  |
| --- |
| describe(**"*Tests* …"**, **function**() {  describe(**"*TODO* …"**, **function**() {  ***it***(**"*TODO …*"**, **function**() {  *//* ***TODO:*** …  });  });  *//* ***TODO:*** …  }); |

The object that should have the following functionality:

* stringSlicer(str) - A function that accepts a single parameter:
  + the function returns the first 3 chars of the str and adds "…" to it
  + if the input string is shorther than 3 chars returns the whole string, and also adds "…" to it
* wordChecker(word, text) - A function that accepts two parameters (strings):
  + the function converts the two strings to lowercase. Trims the spaces for **word** parameter.
  + then the function checks if the word is existing in the text and if so the function returns the word;
  + if the word is not included in the text, the function returns the string: "{word} not found!";
* printEveryNthElement(n, arr) - A function that accepts two parameters (number and array):
  + the function first checks the inputs (if the first parameter is number, and if the second parameter is an array). If one of the parameters is not valid the function throws an Error - **"The input is not valid!"**
  + if both inputs are valid the function loops through the array elements and filters every n-th element
  + then the function returns an array with the filtered elements

### JS Code

To ease you in the process, you are provided with an implementation which meets all of the specification requirements for the **stringOperations** object:

|  |
| --- |
| stringOperations.js |
| const stringOperations = {      stringSlicer: function (str) {          return str.slice(0, 3) + '...';      },      wordChecker: function (word, text) {          word = word.toLowerCase().trim();          text = text.toLowerCase();          if (text.includes(word)) {              return word;          } else {              return `${word} not found!`;          }      },      printEveryNthElement: function (n, arr) {          n = Number(n);          if (isNaN(n) || !Array.isArray(arr)) {              throw new Error('The input is not valid!');          }          const outputArray = [];          for (let i = 0; i < arr.length; i += n) {              outputArray.push(arr[i]);          }          return outputArray;      }  }; |

### Submission

Submit your tests inside a describe() statement, as shown above.

describe('string Operations', () => {

    it('stringSlicer', () => {

        assert.equal(stringOperations.stringSlicer('string'), 'str...');

        assert.equal(stringOperations.stringSlicer('st'), 'st...');

        assert.equal(stringOperations.stringSlicer(['st', 'el2', 'el3']), 'st,el2,el3...');

    });

    it('wordChecker not found', () => {

        assert.equal(stringOperations.wordChecker('Word', 'text'), 'word not found!');

        assert.equal(stringOperations.wordChecker('word', 'tEXt'), 'word not found!');

        assert.equal(stringOperations.wordChecker('WoRd ', 'text'), 'word not found!');

    });

    it('wordChecker if found', () => {

        assert.equal(stringOperations.wordChecker('str', 'string'), 'str');

        assert.equal(stringOperations.wordChecker('stR ', 'stRIng'), 'str');

        assert.equal(stringOperations.wordChecker(' Str', 'striNg'), 'str');

    });

    it('printEveryNthElement', () => {

        assert.throw(() => stringOperations.printEveryNthElement('str', [1,2,3,4]) , 'The input is not valid!');

        assert.throw(() => stringOperations.printEveryNthElement(85, undefined) , 'The input is not valid!');

        assert.throw(() => stringOperations.printEveryNthElement(85, NaN) , 'The input is not valid!');

        assert.throw(() => stringOperations.printEveryNthElement(85, {}) , 'The input is not valid!');

        assert.throw(() => stringOperations.printEveryNthElement(85, 'string') , 'The input is not valid!');

        assert.throw(() => stringOperations.printEveryNthElement(NaN, ['asds', 'asd']) , 'The input is not valid!');

        assert.throw(() => stringOperations.printEveryNthElement(undefined, ['asds', 'asd']) , 'The input is not valid!');

    })

    it('printEveryNthElement valid params', () => {

        assert.deepEqual(stringOperations.printEveryNthElement(2, [1,2,3,4,5]), [1,3,5]);

    })

});