



Js Mastery codewars

study case

If you can't sleep, just count sheep!!

[Instructions :](#)

[Solution](#)

[Sample Tests](#)

studied:

Grasshopper - Terminal game combat function

[Instructions :](#)

[Solution](#)

[Sample Tests](#)

studied:

Convert a Number to a String!

[Instructions :](#)

[Examples \(input --> output\):](#)

[Solution](#)

[Sample Tests](#)

studied:

What is between?

[Instructions :](#)

[Solution](#)

[Sample Tests](#)

studied:

How good are you really?

[Instructions :](#)

Note:

Solution

Sample Tests

studied:

Powers of 2

Instructions :

Examples

Solution

Sample Tests

studied:

Reversing Words in a String

Instructions :

Solution

Sample Tests

studied:

Super Duper Easy

Instructions :

Solution

Sample Tests

studied:

L1: Set Alarm

Instructions :

Solution

Sample Tests

studied:

Multiply

Instructions :

Solution

Sample Tests

studied:

Exclamation marks series #11: Replace all vowel to exclamation mark in the sentence

Instructions :

Examples

Solution

Sample Tests

studied:

Are You Playing Banjo?

Instructions :

[Solution](#)

[Sample Tests](#)

studied:

[Training JS #2: Basic data types--Number](#)

[Instructions :](#)

[Task](#)

[Solution](#)

[Sample Tests](#)

studied:

[Number of Decimal Digits](#)

[Instructions :](#)

[Solution](#)

[Sample Tests](#)

studied:

[Reversed Strings](#)

[Instructions :](#)

[Solution](#)

[Sample Tests](#)

studied:

[Even or Odd](#)

[Instructions :](#)

[Solution](#)

[Sample Tests](#)

studied:

[All Star Code Challenge #18](#)

[Instructions :](#)

[Solution](#)

[Sample Tests](#)

studied:

[Reversed Words](#)

[Instructions :](#)

[Solution](#)

[Sample Tests](#)

studied:

[Ninja vs Samurai: Strike <7 Kyu>](#)

[Instructions :](#)

[Solution](#)

[Sample Tests](#)

studied:

If you can't sleep, just count sheep!!

If you can't sleep, just count sheep!!

Instructions :

Given a non-negative integer, `3` for example, return a string with a murmur: `"1 sheep...2 sheep...3 sheep..."`. Input will always be valid, i.e. no negative integers.

Solution

```
var countSheep = function (num){  
let str = '';  
for(let i = 1; i <= num ; i++){  
str+=  ${i} sheep...;  
}  
return str;  
}
```

Sample Tests

```
const chai = require("chai");  
const assert = chai.assert;  
chai.config.truncateThreshold=0;  
  
describe("Fixed tests", () => {  
it("Testing for fixed tests", () => {  
assert.strictEqual(countSheep(0), "");  
assert.strictEqual(countSheep(1), "1 sheep...");  
assert.strictEqual(countSheep(2), "1 sheep...2 sheep...");  
assert.strictEqual(countSheep(3), "1 sheep...2 sheep...3  
sheep...");  
})});
```

```
});  
});
```

studied:

increment
for
template literal

Grasshopper - Terminal game combat function

Instructions :

Create a combat function that takes the player's current health and the amount of damage received, and returns the player's new health. Health can't be less than 0.

Solution

```
function combat(health, damage) {  
  return Math.max(health - damage, 0);  
}
```

Sample Tests

```
const Test = require('@codewars/test-compat');  
  
describe("The combat() function", function () {  
  it("should work for some example tests", function () {  
    Test.assertEquals(combat(100, 5), 95);  
    Test.assertEquals(combat(92, 8), 84);  
    Test.assertEquals(combat(20, 30), 0, "Health cannot go below 0");  
  });  
});
```

```
0");
});
});
```

studied:

```
module Math
arithmetic
```

Convert a Number to a String!

Instructions :

We need a function that can transform a number (integer) into a string.

What ways of achieving this do you know?

Examples (input --> output):

```
123 --> "123"
999 --> "999"
-100 --> "-100"
```

Solution

```
function numberToString(num) {
return num.toString()
}
```

Sample Tests

```
const assert = require('chai').assert;
describe("Tests", () => {
  it("test", () => {
    assert.strictEqual(numberToString(67), '67');
  });
});
```

studied:

typeof
arithmetic

What is between?

Instructions :

Complete the function that takes two integers (`a, b`, where `a < b`) and return an array of all integers between the input parameters, **including** them.

For example:

```
a = 1
b = 4
--> [1, 2, 3, 4]
```

Solution

```
function between(a, b) {
  const betweenArray = []
  while (a <= b) {
```

```
betweenArray.push(a);
a++;
}

return betweenArray;
}
```

Sample Tests

```
describe("Basic tests", () => {
  it("between(1, 4)", () => assert.deepStrictEqual(between(1, 4),
[1, 2, 3, 4]));
  it("between(-2, 2)", () => assert.deepStrictEqual(between(-2,
2), [-2, -1, 0, 1, 2]));
});
```

studied:

typeof
arithmetic

How good are you really?

Instructions :

There was a test in your class and you passed it.
Congratulations! But you're an ambitious person. You want to know if you're better than the average student in your class.
You receive an array with your peers' test scores. Now calculate the average and compare your score!
Return `True` if you're better, else `False`!

Note:

Your points are not included in the array of your class's points. For calculating the average point you may add your point to the given array!

Solution

```
function betterThanAverage(classPoints, yourPoints) {  
    return yourPoints > classPoints.reduce((a, b) => a + b) /  
        classPoints.length;  
}
```

Sample Tests

```
const chai = require('chai');  
const assert = chai.assert;  
  
describe("Example Tests", function() {  
    it("betterThanAverage([2, 3], 5) should return True", function() {  
        assert.strictEqual(betterThanAverage([2, 3], 5), true);  
    });  
  
    it("betterThanAverage([100, 40, 34, 57, 29, 72, 57, 88], 75)  
    should return True", function() {  
        assert.strictEqual(betterThanAverage([100, 40, 34, 57, 29, 72,  
        57, 88], 75), true);  
    });  
  
    it("betterThanAverage([12, 23, 34, 45, 56, 67, 78, 89, 90], 9)  
    should return False", function() {  
        assert.strictEqual(betterThanAverage([12, 23, 34, 45, 56, 67,  
        78, 89, 90], 9), false);  
    });
```

```

it("betterThanAverage([41, 75, 72, 56, 80, 82, 81, 33], 50)
should return False", function() {
assert.strictEqual(betterThanAverage([41, 75, 72, 56, 80, 82,
81, 33], 50), false);
});

it("betterThanAverage([29, 55, 74, 60, 11, 90, 67, 28], 21)
should return False", function() {
assert.strictEqual(betterThanAverage([29, 55, 74, 60, 11, 90,
67, 28], 21), false);
});
});

```

studied:

typeof
arithmetic

Powers of 2

Instructions :

Complete the function that takes a non-negative integer `n` as input, and returns a list of all the powers of `2` with the exponent ranging from `0` to `n` (inclusive).

Examples

```

n = 0  ==> [1]      # [2^0]
n = 1  ==> [1, 2]    # [2^0, 2^1]
n = 2  ==> [1, 2, 4] # [2^0, 2^1, 2^2]

```

Solution

```
function powersOfTwo(n){  
var result = [];  
for (var i = 0; i <= n; i++) {  
result.push(Math.pow(2, i));  
}  
return result;  
}
```

Sample Tests

```
const chai = require("chai");  
const assert = chai.assert;  
chai.config.truncateThreshold=0;  
  
describe("Basic Tests", function(){  
it("Testing for fixed tests", () => {  
assert.deepEqual(powersOfTwo(0), [1])  
assert.deepEqual(powersOfTwo(1), [1, 2])  
assert.deepEqual(powersOfTwo(4), [1, 2, 4, 8, 16])  
})  
});
```

studied:

typeof
arithmetic

Reversing Words in a String

Instructions :

You need to write a function that reverses the words in a given string. A word can also fit an empty string. If this is not clear enough, here are some examples:

As the input may have trailing spaces, you will also need to ignore unnecessary whitespace.

Example (**Input** --> **Output**)

```
"Hello World" --> "World Hello"  
"Hi There." --> "There. Hi"
```

Happy coding!

Solution

```
function combat(health, damage) {  
  return Math.max(health - damage, 0);  
}
```

Sample Tests

```
const Test = require('@codewars/test-compat');  
  
describe("The combat() function", function () {  
  it("should work for some example tests", function () {  
    Test.assertEquals(combat(100, 5), 95);  
    Test.assertEquals(combat(92, 8), 84);  
    Test.assertEquals(combat(20, 30), 0, "Health cannot go below  
    0");  
  });  
});
```

studied:

```
module Math  
arithmetic
```

Super Duper Easy

Instructions :

Make a function that returns the value multiplied by 50 and increased by 6. If the value entered is a string it should return "Error".

Solution

```
function problem(x){  
return typeof x === "number" ? x * 50 + 6 : "Error"  
}
```

Sample Tests

```
const chai = require("chai");  
const assert = chai.assert;  
chai.config.truncateThreshold=0;  
  
describe("Basic tests", () => {  
it("Testing for fixed tests", () => {  
assert.strictEqual(problem("hello"), "Error");  
assert.strictEqual(problem(1), 56);  
assert.strictEqual(problem(5), 256);  
assert.strictEqual(problem(0), 6);  
assert.strictEqual(problem(1.2), 66);  
assert.strictEqual(problem(3), 156);
```

```
assert.strictEqual(problem("RyanIsCool"), "Error");
assert.strictEqual(problem(-3), -144);
assert.strictEqual(problem(""), "Error");
assert.strictEqual(problem(0.03), 7.5);
})
})
```

studied:

typeof
arithmetic

L1: Set Alarm

Instructions :

Write a function named `setAlarm` which receives two parameters. The first parameter, `employed`, is true whenever you are employed and the second parameter, `vacation` is true whenever you are on vacation.

The function should return true if you are employed and not on vacation (because these are the circumstances under which you need to set an alarm). It should return false otherwise.

Examples:

```
setAlarm(true, true) -> false
setAlarm(false, true) -> false
setAlarm(false, false) -> false
setAlarm(true, false) -> true
```

Solution

```
function setAlarm(employed, vacation){  
    return employed && !vacation;  
}
```

Sample Tests

```
const chai = require("chai");  
const assert = chai.assert;  
chai.config.truncateThreshold=0;  
  
describe("Test Suite", ()=>{  
    it("Fixed tests", ()=>{  
        assert.strictEqual(setAlarm(true, true), false, "Should be  
        false.");  
        assert.strictEqual(setAlarm(false, true), false, "Should be  
        false.");  
        assert.strictEqual(setAlarm(true, false), true, "Should be  
        true.");  
    });  
});
```

studied:

typeof
arithmetic

Multiply

Instructions :

This code does not execute properly. Try to figure out why.

Solution

```
function multiply(a, b){  
    return a * b;  
}
```

Sample Tests

```
const assert = require("chai").assert;  
  
describe("Multiply", () => {  
    it("fixed tests", () => {  
        assert.strictEqual(multiply(1,1), 1);  
        assert.strictEqual(multiply(2,1), 2);  
        assert.strictEqual(multiply(2,2), 4);  
        assert.strictEqual(multiply(3,5), 15);  
    });  
});
```

studied:

typeof
arithmetic

**Exclamation marks series #11:
Replace all vowel to exclamation
mark in the sentence**

Instructions :

Replace all vowel to exclamation mark in the sentence.
aeiouAEIOU
is vowel.

Examples

```
replace("Hi!") === "H!!"  
replace("!Hi! Hi!") === "!H!! H!!"  
replace("aeiou") === "!!!!!"  
replace("ABCDE") === "!BCD!"
```

Solution

```
function replace(s){  
return s.replace(/[aeiou]/ig, '!');  
}
```

Sample Tests

```
const Test = require('@codewars/test-compat');  
  
describe("Basic Tests", function(){  
it("It should works for basic tests", function(){  
  
Test.assertSimilar(replace("Hi!") , "H!!")  
Test.assertSimilar(replace("!Hi! Hi!") , "!H!! H!!")  
Test.assertSimilar(replace("aeiou") , "!!!!!")  
Test.assertSimilar(replace("ABCDE") , "!BCD!")  
  
}))})
```

studied:

typeof

Are You Playing Banjo?

Instructions :

Create a function which answers the question "Are you playing banjo?". If your name starts with the letter "R" or lower case "r", you are playing banjo!

The function takes a name as its only argument, and returns one of the following strings:

```
name + " plays banjo"  
name + " does not play banjo"
```

Names given are always valid strings.

Solution

```
function areYouPlayingBanjo(name) {  
  return name + (name[0].toLowerCase() == 'r' ? ' plays' : ' does  
not play') + ' banjo';  
}
```

Sample Tests

```
const chai = require("chai");  
const assert = chai.assert;  
chai.config.truncateThreshold=0;  
  
describe("Basic tests", () => {  
  it("Testing for fixed tests", () => {  
    assert.strictEqual(areYouPlayingBanjo("Adam"), "Adam does not
```

```
play banjo");
assert.strictEqual(areYouPlayingBanjo("Paul"), "Paul does not
play banjo");
assert.strictEqual(areYouPlayingBanjo("Ringo"), "Ringo plays
banjo");
assert.strictEqual(areYouPlayingBanjo("bravo"), "bravo does not
play banjo");
assert.strictEqual(areYouPlayingBanjo("rolf"), "rolf plays
banjo");
})
})
```

studied:

typeof
arithmetic

Training JS #2: Basic data types-- Number

Instructions :

In javascript, Number is one of basic data types. It can be positive: `1, 2, 3`, negative: `-1, -100`, integer: `123, 456`, decimal: `3.1415926, -8.88` etc..

Numbers can use operators such as `+ - * / %`

Task

I've written five function `equal1, equal2, equal3, equal4, equal5`, defines six global variables `v1 v2 v3 v4 v5 v6`, every function has two local variables `a, b`, please set the appropriate value for the two variables(select from v1--v6), making these function return

value equal to 100. the function equal1 is completed, please refer to this example to complete the following functions.

When you have finished the work, click "Run Tests" to see if your code is working properly.

In the end, click "Submit" to submit your code pass this kata.

Solution

```
let v1=50,  
v2=100,  
v3=150,  
v4=200,  
v5=2,  
v6=250;  
  
const equal1 = () => v1 + v1;  
const equal2 = () => v3 - v1;  
const equal3 = () => v1 * v5;  
const equal4 = () => v4 / v5;  
const equal5 = () => v2 % v4;
```

Sample Tests

```
const { assert } = require('chai');  
  
describe("Tests", () => {  
  it("test", () => {  
    assert.strictEqual(equal1(), 100, "value of a+b is not equal to  
    100");  
    assert.strictEqual(equal2(), 100, "value of a-b is not equal to  
    100");  
    assert.strictEqual(equal3(), 100, "value of a*b is not equal to  
    100");  
  });  
});
```

```
assert.strictEqual(equal4(), 100, "value of a/b is not equal to 100");
assert.strictEqual(equal5(), 100, "value of a%b is not equal to 100");
});
});
```

studied:

typeof
arithmetic

Number of Decimal Digits

Instructions :

Determine the total number of digits in the integer (`n>=0`) given as input to the function. For example, 9 is a single digit, 66 has 2 digits and 128685 has 6 digits. Be careful to avoid overflows/underflows.

All inputs will be valid.

Solution

```
function digits(n) {
return n.toString().length;
}
```

Sample Tests

```
describe("Solution", function(){
  it("Example tests", function(){
    Test.assertEquals(digits(5), 1, "Fail!");
    Test.assertEquals(digits(12345), 5, "Fail!");
    Test.assertEquals(digits(9876543210), 10, "Fail!");
  });
});
```

studied:

typeof
arithmetic

Reversed Strings

Instructions :

Complete the solution so that it reverses the string passed into it.

```
'world'  => 'dlrow'  
'word'   => 'drow'
```

Solution

```
function solution(str){  
  return str.split('').reverse().join('');  
}
```

Sample Tests

```
const chai = require("chai");
const assert = chai.assert;
chai.config.truncateThreshold=0;

describe("Basic tests", () => {
  it("Testing for fixed tests", () => {
    assert.strictEqual(solution('world'), 'dlrow');
    assert.strictEqual(solution('hello'), 'olleh');
    assert.strictEqual(solution(''), '');
    assert.strictEqual(solution('h'), 'h');
  });
});
```

studied:

typeof
arithmetic

Even or Odd

Instructions :

Create a function that takes an integer as an argument and returns "Even" for even numbers or "Odd" for odd numbers.

Solution

```
function even_or_odd(number) {
  return number % 2 ? "Odd" : "Even"
}
```

Sample Tests

```
const chai = require('chai');
const assert = chai.assert;

describe("Sample tests", () => {

  it("2 is even", () => {
    assert.strictEqual(evenOrOdd(2), "Even");
  });
  it("7 is odd", () => {
    assert.strictEqual(evenOrOdd(7), "Odd");
  });
  it("-42 is even", () => {
    assert.strictEqual(evenOrOdd(-42), "Even");
  });
  it("-7 is odd", () => {
    assert.strictEqual(evenOrOdd(-7), "Odd");
  });
  it("0 is even", () => {
    assert.strictEqual(evenOrOdd(0), "Even");
  });
});
```

studied:

typeof
arithmetic

All Star Code Challenge #18

Instructions :

This Kata is intended as a small challenge for my students

All Star Code Challenge #18

Create a function that accepts 2 string arguments and returns an integer of the count of occurrences the 2nd argument is found in the first one.

If no occurrences can be found, a count of 0 should be returned.

```
("Hello", "o")  ==> 1
("Hello", "l")  ==> 2
("", "z")       ==> 0
```

Notes:

- The first argument can be an empty string
- The second string argument will always be of length 1

Solution

```
function strCount(str, letter){
  return str.split(letter).length-1
}
```

Sample Tests

```
const { assert } = require('chai');

describe("Tests", () => {
  it("test", () => {
    assert.strictEqual(strCount('Hello', 'o'), 1);
    assert.strictEqual(strCount('Hello', 'l'), 2);
    assert.strictEqual(strCount('', 'z'), 0);
  });
});
```

studied:

typeof

arithmetic

Reversed Words

Instructions :

Complete the solution so that it reverses all of the words within the string passed in.

Words are separated by exactly one space and there are no leading or trailing spaces.

Example(Input --> Output):

```
"The greatest victory is that which requires no battle" --> "battle no requires which that  
is victory greatest The"
```

Solution

```
function reverseWords(str){  
  return str.split(' ').reverse().join(' ');  
}
```

Sample Tests

```
const { assert } = require('chai');  
  
describe("reverseWords", function(){  
  it("should work for some examples", function(){  
    assert.strictEqual(reverseWords("hello world!")
```

```
), "world! hello")
assert.strictEqual(reverseWords("yoda doesn't speak like this"
), "this like speak doesn't yoda")
assert.strictEqual(reverseWords("foobar"
), "foobar")
assert.strictEqual(reverseWords("kata editor"
), "editor kata")
assert.strictEqual(reverseWords("row row row your boat"
), "boat your row row row")
assert.strictEqual(reverseWords("")
), "")
});
});
```

studied:

typeof
arithmetic

Ninja vs Samurai: Strike <7 Kyu>

Instructions :

Something is wrong with our Warrior class. The strike method does not work correctly. The following shows an example of this code being used:

```
var ninja = new Warrior('Ninja');
var samurai = new Warrior('Samurai');

samurai.strike(ninja, 3);
// ninja.health should == 70
```

Can you figure out what is wrong?

Solution

```
function Warrior(name){  
  this.name  
  = name;  
  this.health = 100;  
  this.strike = (enemy, swings) => enemy.health = Math.max(0,  
  enemy.health - (Math.abs(swings) * 10));  
}
```

Sample Tests

```
// Since Node 10, we're using Mocha.  
// You can use
```

```
chai
```

```
for assertions.  
const chai = require("chai");  
const assert = chai.assert;  
// Uncomment the following line to disable truncating failure  
messages for deep equals, do:  
// chai.config.truncateThreshold = 0;  
// Since Node 12, we no longer include assertions from our  
deprecated custom test framework by default.  
// Uncomment the following to use the old assertions:  
// const Test = require("@codewars/test-compat");  
  
describe("Solution", function() {  
  it("should test for something", function() {  
    // Test.assertEquals(1 + 1, 2);  
    // assert.strictEqual(1 + 1, 2);  
  });  
});
```

studied:

typeof

arithmetic