# Solutions-Lab: Unit Testing with Mocha

# Error Handling

## 1.Sub Sum

function takeSubSum(arr, startIndex, endIndex){

// if (!Array.isArray(arr)){

// return NaN;

// }

if (startIndex < 0){

startIndex = 0;

}

if (endIndex > arr.length - 1){

endIndex = arr.length - 1;

}

let sum = 0;

for (let i = startIndex; i <= endIndex; i++) {

sum += Number(arr[i]);

}

return sum;

}

console.log(takeSubSum([10, 20, 30, 40, 50, 60], 3, 300))

console.log(takeSubSum([1.1, 2.2, 3.3, 4.4, 5.5], -3, 1))

console.log(takeSubSum([10, 'twenty', 30, 40], 0, 2))

console.log(takeSubSum([], 1, 2))

console.log(takeSubSum('text', 0, 2))

|  |
| --- |
| function subSum(arr, startIndex, endIndex) { |
|  | if(! Array.isArray(arr)){ |
|  | return NaN; |
|  | } |
|  | if(startIndex < 0){ |
|  | startIndex = 0; |
|  | } |
|  | if(endIndex > arr.length - 1){ |
|  | endIndex = arr.length - 1; |
|  | } |
|  |  |
|  | let sum = 0; |
|  |  |
|  | for(let i = startIndex; i <= endIndex; i++){ |
|  | sum += Number(arr[i]); |
|  | } |
|  |  |
|  | return sum; |
|  | } |

## 2.Playing Cards

function makeCard(cardFace, cardSuit){

let cardFaces = ['2', '3', '4', '5', '6', '7', '8', '9', '10', 'J', 'Q', 'K', 'A'];

let cardSuits = {

'S': '\u2660',

'H': '\u2665',

'D': '\u2666',

'C': '\u2663'

};

if (!cardFaces.includes(cardFace)){

throw new Error('Invalid card face: ' + cardFace);

}

if (!Object.keys(cardSuits).includes(cardSuit)){

throw new Error('Invalid card suit: ' + cardSuit);

}

let card = {

face: cardFace,

suit: cardSuits[cardSuit],

toString: () => `${card.face}${card.suit}`

};

// let card = {

// toString: () => `${cardFace}${cardSuits[cardSuit]}`

// };

//return `${cardFace}${cardSuits[cardSuit]}`;

return card;

}

console.log('' + makeCard('A', 'S'))

console.log('' + makeCard('10', 'H'))

console.log('' + makeCard('1', 'C'))

|  |
| --- |
| function makeCard(cardFace, cardSuit) { |
|  | const Faces = ["2", "3", "4", "5", "6", "7", "8", "9", "10", "J", "Q", "K", "A"]; |
|  | const Suits = { |
|  | "S": "\u2660", |
|  | "H": "\u2665", |
|  | "D": "\u2666", |
|  | "C": "\u2663" |
|  | }; |
|  |  |
|  | if (!Faces.includes(cardFace)) |
|  | throw new Error("Invalid card face: " + cardFace); |
|  | if (!Object.keys(Suits).includes(cardSuit)) |
|  | throw new Error("Invalid card suit: " + cardSuit); |
|  |  |
|  |  |
|  | let card = { |
|  | face: cardFace, |
|  | suit: Suits[cardSuit], |
|  | toString: () => `${card.face}${card.suit}` |
|  | } |
|  | return card; |
|  | } |
|  |  |
|  | // console.log('' + makeCard('A', 'S')); |
|  | console.log('' + makeCard('J', 'D')); |

## 3.Deck of Cards

function printDeckOfCards(cards){

let deckOfCards = [];

for(let card of cards){

let cardFace = card.substring(0, card.length - 1);

let cardSuit = card.substring(card.length - 1);

try {

deckOfCards.push(makeCard(cardFace, cardSuit).toString());

}

catch(e){

console.log('Invalid card: ' + card);

return;

//throw new Error('Invalid card: ' + card);

}

}

console.log(deckOfCards.join(' '));

function makeCard(cardFace, cardSuit){

let cardFaces = ['2', '3', '4', '5', '6', '7', '8', '9', '10', 'J', 'Q', 'K', 'A'];

let cardSuits = {

'S': '\u2660',

'H': '\u2665',

'D': '\u2666',

'C': '\u2663'

};

if (!cardFaces.includes(cardFace)){

throw new Error('Invalid card face: ' + cardFace);

}

if (!Object.keys(cardSuits).includes(cardSuit)){

throw new Error('Invalid card suit: ' + cardSuit);

}

let card = {

face: cardFace,

suit: cardSuits[cardSuit],

toString: () => `${card.face}${card.suit}`

};

// let card = {

// toString: () => `${cardFace}${cardSuits[cardSuit]}`

// };

//return `${cardFace}${cardSuits[cardSuit]}`;

return card;

}

}

printDeckOfCards(['AS', '10D', 'KH', '2C'])

printDeckOfCards(['5S', '3D', 'QD', '1C'])

|  |
| --- |
| function deckOfCards(cards) { |
|  | class Card{ |
|  | constructor(face, suit){ |
|  | this.face = face; |
|  | this.suit = suit; |
|  | } |
|  |  |
|  | get face(){ |
|  | return this.\_face; |
|  | } |
|  | set face(newFace){ |
|  | const validFaces = ['2', '3', '4', '5', '6', '7', '8', '9', '10', 'J', 'Q', 'K', 'A']; |
|  | if(! validFaces.includes(newFace)){ |
|  | throw new Error("Invalid card face: " + newFace); |
|  | } |
|  |  |
|  | this.\_face = newFace; |
|  | } |
|  |  |
|  | get suit(){ |
|  | return this.\_suit; |
|  | } |
|  | set suit(newSuit){ |
|  | const validSuits = ['C', 'D', 'H', 'S']; |
|  | if(! validSuits.includes(newSuit)){ |
|  | throw new Error("Invalid card suit: " + newSuit); |
|  | } |
|  |  |
|  | this.\_suit = newSuit; |
|  | } |
|  |  |
|  | toString(){ |
|  | let suitToChar = { |
|  | 'C': "\u2663", |
|  | 'D': "\u2666", |
|  | 'H': "\u2665", |
|  | 'S': "\u2660" |
|  | }; |
|  |  |
|  | return this.face + suitToChar[this.suit]; |
|  | } |
|  | } |
|  |  |
|  | let deck = []; |
|  | for(let cardStr of cards){ |
|  | let face = cardStr.substring(0, cardStr.length - 1); |
|  | let suit = cardStr.substr(cardStr.length - 1, 1); |
|  |  |
|  | try { |
|  | deck.push(new Card(face, suit)); |
|  | } catch(err) { |
|  | console.log("Invalid card: " + cardStr); |
|  | return; |
|  | } |
|  | } |
|  |  |
|  | console.log(deck.join(" ")); |
|  |  |

|  |
| --- |
| function printDeckOfCards(cards) { |
|  | function makeCard(cardFace, cardSuit) { |
|  | const Faces = ["2", "3", "4", "5", "6", "7", "8", "9", "10", "J", "Q", "K", "A"]; |
|  | const Suits = { |
|  | "S": "\u2660", |
|  | "H": "\u2665", |
|  | "D": "\u2666", |
|  | "C": "\u2663" |
|  | }; |
|  |  |
|  | if (!Faces.includes(cardFace)) |
|  | throw new Error("Invalid card face: " + cardFace); |
|  | if (!Object.keys(Suits).includes(cardSuit)) |
|  | throw new Error("Invalid card suit: " + cardSuit); |
|  |  |
|  |  |
|  | let card = { |
|  | face: cardFace, |
|  | suit: Suits[cardSuit], |
|  | toString: () => `${card.face}${card.suit}` |
|  | } |
|  | return card; |
|  | } |
|  |  |
|  | let result = []; |
|  | for (let i = 0; i < cards.length; i++) { |
|  | const element = cards[i]; |
|  |  |
|  | let face = element.substr(0, element.length - 1); |
|  | let suit = element.substr(element.length - 1, 1); |
|  | try { |
|  | result.push(makeCard(face, suit).toString()); |
|  | } catch (ex) { |
|  | console.log(`Invalid card: ${element}`); |
|  | return; |
|  | } |
|  |  |
|  | } |
|  | console.log(result.join(" ")); |
|  | } |
|  |  |
|  | printDeckOfCards(['AS', '10D', 'KH', '2C']); |
|  | printDeckOfCards(['5S', '3D', 'QD', '1C']); |

# Unit Testing

## 4.Sum of Numbers

function sum(arr) {

let sum = 0;

for (num of arr)

sum += Number(num);

return sum;

}

module.exports = {sum};

let expect = require('chai').expect;

let sum = require('../04SumOfNumbers.js').sum;

//In Judge must be paste without this above

describe('sum(arr) - sum array of numbers', function() {

it('should return 3 for [1, 2]', function() {

expect(sum([1, 2])).to.be.equal(3);

});

it('should return 100101 for [1, 100000, 100]', function(){

expect(sum([1, 100000, 100])).to.be.equal(100101);

});

it('should return -7 for [10, 0, -17]', function(){

expect(sum([10, 0, -17])).to.be.equal(-7);

});

it('should return 1.7 for [1, 0, 0.7]', function(){

expect(sum([1, 0, 0.7])).to.be.equal(1.7);

});

it('should return 10 for [10]', function(){

expect(sum([10])).to.be.equal(10);

});

it('should return 0 for empty array', function(){

expect(sum([])).to.be.equal(0);

});

it('should return NaN for invalid data', function(){

expect(sum(['invalid data'])).to.be.NaN;

});

it('should return NaN for not array', function(){

expect(sum('invalid data')).to.be.NaN;

});

});

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | | function sum(arr) { | |  | let sum = 0; | |  | for (num of arr) | |  | sum += Number(num); | |  | return sum; | |  | } | |  |  | |  | module.exports = { sum }; |   let expect = require("chai").expect; |
|  | let sum = require("../01. Sum of Numbers").sum; |
|  |  |
|  | describe("sum(arr) - sum array of numbers", function () { |
|  | it("should return 3 for [1,2]", function () { |
|  | expect(sum([1,2])).to.be.equal(3); |
|  | }); |
|  | it("should return 1 for [1]", function () { |
|  | expect(sum([1])).to.be.equal(1); |
|  | }); |
|  | it("should return 0 for empty array", function () { |
|  | expect(sum([])).to.be.equal(0); |
|  | }); |
|  | it("should return 3 for [1.5, 2.5, -1]", function () { |
|  | expect(sum([1.5, 2.5, -1])).to.be.equal(3); |
|  | }); |
|  | it("should return NaN for invalid data", function () { |
|  | expect(sum("invalid data")).to.be.NaN; |
|  | }); |
|  | }); |

|  |
| --- |
| function sum(arr) { |
|  | let sum = 0; |
|  | for (num of arr) |
|  | sum += Number(num); |
|  | return sum; |
|  | } |
|  |  |
|  | module.exports = { |
|  | sum |
|  | }; |

|  |
| --- |
| let expect = require("chai").expect; |
|  | let sum = require("../4\_SumOfNumbers.js").sum; |
|  |  |
|  | describe("sum(arr)", function () { |
|  | it("should return 3 for [1,2]", function () { |
|  | expect(sum([1, 2])).to.be.equal(3); |
|  | }); |
|  | it("should return 5 for [3,2]", function () { |
|  | expect(sum([3, 2])).to.be.equal(5); |
|  | }); |
|  | it("should return 0 for [0,0]", function () { |
|  | expect(sum([0, 0])).to.be.equal(0); |
|  | }); |
|  | it("should return 100 for [99,1]", function () { |
|  | expect(sum([99, 1])).to.be.equal(100); |
|  | }); |
|  | }); |

## 5.Check for Symmetry

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;

}

module.exports = {isSymmetric};

let expect = require('chai').expect;

let isSymmetric = require('../05CheckForSymmetry.js').isSymmetric;

//In Judge must be paste without this above

describe('isSymmetric(arr) - checks if the array is symmetric', function(){

it('should return true for [1, 2, 3, 3, 2, 1]', function(){

expect(isSymmetric([1, 2, 3, 3, 2, 1])).to.be.equal(true);

});

it('should return true for [1, 2, 3, 2, 1]', function(){

expect(isSymmetric([1, 2, 3, 2, 1])).to.be.equal(true);

});

it('should return true for [1]', function(){

expect(isSymmetric([1])).to.be.equal(true);

});

it('should return true for ["hi"]', function(){

expect(isSymmetric(["hi"])).to.be.equal(true);

});

it('should return true for ["a", "bb", "ccc", "ccc", "bb", "a"]', function(){

expect(isSymmetric(["a", "bb", "ccc", "ccc", "bb", "a"])).to.be.equal(true);

});

it('should return true for ["a", "bb", "ccc", "bb", "a"]', function(){

expect(isSymmetric(["a", "bb", "ccc", "bb", "a"])).to.be.equal(true);

});

it('should return true for ["a", 1, new Date("2017-01-10"), {c: "zy"}, "ccc", {c: "zy"}, new Date("2017-01-10"), 1, "a"]', function(){

expect(isSymmetric(["a", 1, new Date("2017-01-10"), {c: "zy"}, "ccc", {c: "zy"}, new Date("2017-01-10"), 1, "a"])).to.be.equal(true);

});

it('should return false for [1, 2, 3]', function(){

expect(isSymmetric([1, 2, 3])).to.be.equal(false);

});

it('should return false for ["aba", "bab", "bab"]', function(){

expect(isSymmetric(["aba", "bab", "bab"])).to.be.equal(false);

});

it('should return false for ["a", 1, new Date("2017-01-10"), {c: "zy"}, "ccc", {c: "ZY"}, new Date("2017-01-10"), 1, "a"]', function(){

expect(isSymmetric(["a", 1, new Date("2017-01-10"), {c: "zy"}, "ccc", {c: "ZY"}, new Date("2017-01-10"), 1, "a"])).to.be.equal(false);

});

it('should return false for {"[1, 2, 1]" : "[1, 2, 1]"}', function(){

expect(isSymmetric({"[1, 2, 1]" : "[1, 2, 1]"})).to.be.equal(false);

});

it('should return false for "1, 2, 1"', function(){

expect(isSymmetric("1, 2, 1")).to.be.equal(false);

});

it('should return false for 1, 2, 1', function(){

expect(isSymmetric(1, 2, 1)).to.be.equal(false);

});

it('should return false for "1"', function(){

expect(isSymmetric("1")).to.be.equal(false);

});

it('should return false for 1', function(){

expect(isSymmetric(1)).to.be.equal(false);

});

// it('should return false for []', function(){ //this is wrong for Judge

// expect(isSymmetric([])).to.be.equal(false);

// });

it('should return false for true', function(){

expect(isSymmetric(true)).to.be.equal(false);

});

it('should return false for false', function(){

expect(isSymmetric(false)).to.be.equal(false);

});

});

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | | 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; | |  | } | |  |  | |  | module.exports = { isSymmetric }; |   let expect = require('chai').expect; |
|  | let isSymmetric = require("../02. Check for Symmetry").isSymmetric; |
|  |  |
|  | describe("isSymmetric(arr)", function () { |
|  | it("should return true for [1,2,3,3,2,1]", function () { |
|  | expect(isSymmetric([1,2,3,3,2,1])).to.be.equal(true); |
|  | }); |
|  | it("should return false for [1,2,3,4,2,1]", function () { |
|  | expect(isSymmetric([1,2,3,4,2,1])).to.be.equal(false); |
|  | }); |
|  | it("should return true for [-1,2,-1]", function () { |
|  | expect(isSymmetric([-1,2,-1])).to.be.equal(true); |
|  | }); |
|  | it("should return false for [-1,2,1]", function () { |
|  | expect(isSymmetric([-1,2,1])).to.be.equal(false); |
|  | }); |
|  | it("should return false for [1,2]", function () { |
|  | expect(isSymmetric([1,2])).to.be.equal(false); |
|  | }); |
|  | it("should return true for [1]", function () { |
|  | expect(isSymmetric([1])).to.be.equal(true); |
|  | }); |
|  | it("should return true for [5,'hi',{a:5},new Date(),{a:5},'hi',5]", function () { |
|  | expect(isSymmetric([5,'hi',{a:5},new Date(),{a:5},'hi',5])).to.be.equal(true); |
|  | }); |
|  | it("should return false for [5,'hi',{a:5},new Date(),{X:5},'hi',5]", function () { |
|  | expect(isSymmetric([5,'hi',{a:5},new Date(),{X:5},'hi',5])).to.be.equal(false); |
|  | }); |
|  | it("should return false for 1,2,2,1", function () { |
|  | expect (isSymmetric(1,2,2,1)).to.be.equal(false); |
|  | }); |
|  | }); |

|  |
| --- |
| 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; |
|  | } |
|  |  |
|  | module.exports = { |
|  | isSymmetric |
|  | }; |

|  |
| --- |
| let expect = require("chai").expect; |
|  | let isSymmetric = require("../5\_CheckForSymmetry.js").isSymmetric; |
|  |  |
|  | describe("isSymmetric", function () { |
|  | it("should return false for non-array {}", function () { |
|  | expect(isSymmetric({})).to.be.equal(false); |
|  | }); |
|  | it("should return false for non-array 1 2 3", function () { |
|  | expect(isSymmetric(1, 2, 3)).to.be.equal(false); |
|  | }); |
|  | it("should return false for non-array hello", function () { |
|  | expect(isSymmetric("hello")).to.be.equal(false); |
|  | }); |
|  | it("should return false for non-symmetryc array", function () { |
|  | expect(isSymmetric([1, 1, 0])).to.be.equal(false); |
|  | }); |
|  | it("should return false for non-symmetryc array", function () { |
|  | expect(isSymmetric([1, 2, 3, 4, 2, 1])).to.be.equal(false); |
|  | }); |
|  | it("should return false for non-symmetryc array", function () { |
|  | expect(isSymmetric([-1, 2, 1])).to.be.equal(false); |
|  | }); |
|  | it("should return false for non-symmetryc array", function () { |
|  | expect(isSymmetric([1, 2])).to.be.equal(false); |
|  | }); |
|  | it("should return true for symmetryc array", function () { |
|  | expect(isSymmetric([1, 1, 1])).to.be.equal(true); |
|  | }); |
|  | it("should return true for symmetryc array", function () { |
|  | expect(isSymmetric([1, 0, 1])).to.be.equal(true); |
|  | }); |
|  | it("should return true for symmetryc array", function () { |
|  | expect(isSymmetric([0, 1, 0])).to.be.equal(true); |
|  | }); |
|  | it("should return true for symmetryc array", function () { |
|  | expect(isSymmetric([0])).to.be.equal(true); |
|  | }); |
|  | it("should return true for symmetryc array", function () { |
|  | expect(isSymmetric([5, 'hi', { |
|  | a: 5 |
|  | }, new Date(), { |
|  | a: 5 |
|  | }, 'hi', 5])).to.be.equal(true); |
|  | }); |
|  | it("should return true for symmetryc array", function () { |
|  | expect(isSymmetric([5, 'hi', { |
|  | a: 5 |
|  | }, new Date(), { |
|  | X: 5 |
|  | }, 'hi', 5])).to.be.equal(false); |
|  | }); |
|  | }) |

## 6.RGB to Hex

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;

}

module.exports = {isSymmetric};

let expect = require('chai').expect;

let rgbToHexColor = require('../06RGBtoHex.js').rgbToHexColor;

//In Judge must be paste without this above

describe('rgbToHexColor(red, green, blue) - converts rgb decimal value to hexadecimal', function(){

describe('Normal cases(valid input)', function(){

it('shuld return #000000 for (0, 0, 0)', function(){

expect(rgbToHexColor(0, 0, 0)).to.be.equal('#000000');

});

it('shuld return #FFFFFF for (255, 255, 255)', function(){

expect(rgbToHexColor(255, 255, 255)).to.be.equal('#FFFFFF');

});

it('shuld return #FF9EAA for (255, 158, 170)', function(){

expect(rgbToHexColor(255, 158, 170)).to.be.equal('#FF9EAA');

});

it('shuld return #FF9EAA for (255, 158, 170)', function(){

expect(rgbToHexColor(255, 158, 170)).to.be.equal('#FF9EAA');

});

it('shuld return #0C0D0E for (12, 13, 14)', function(){

expect(rgbToHexColor(12, 13, 14)).to.be.equal('#0C0D0E');

});

});

describe('Special cases(invalid input)', function(){

it('should return undefined for (10, 10, -10)', function(){

expect(rgbToHexColor(10, 10, -10)).to.be.equal(undefined);

});

it('should return undefined for (16, -10, 10)', function(){

expect(rgbToHexColor(16, -10, 10)).to.be.equal(undefined);

});

it('should return undefined for (-10, 16, 10)', function(){

expect(rgbToHexColor(-10, 16, 10)).to.be.equal(undefined);

});

it('should return undefined for (256, 10, 10)', function(){

expect(rgbToHexColor(256, 10, 10)).to.be.equal(undefined);

});

it('should return undefined for (10, 256, 10)', function(){

expect(rgbToHexColor(10, 256, 10)).to.be.equal(undefined);

});

it('should return undefined for (10, 10, 256)', function(){

expect(rgbToHexColor(10, 10, 256)).to.be.equal(undefined);

});

it('should return undefined for ("k", 16, 10)', function(){

expect(rgbToHexColor("k", 16, 10)).to.be.equal(undefined);

});

it('should return undefined for (10, "k", 10)', function(){

expect(rgbToHexColor(10, "k", 10)).to.be.equal(undefined);

});

it('should return undefined for (10, 10, "k")', function(){

expect(rgbToHexColor(10, 10, "k")).to.be.equal(undefined);

});

it('should return undefined for (10, 10, [10])', function(){

expect(rgbToHexColor(10, 10, [10])).to.be.equal(undefined);

});

it('should return undefined for (10, [10], 10)', function(){

expect(rgbToHexColor(10, [10], 10)).to.be.equal(undefined);

});

it('should return undefined for ([10], 10, 10)', function(){

expect(rgbToHexColor([10], 10, 10)).to.be.equal(undefined);

});

it('should return undefined for (10, 10, [])', function(){

expect(rgbToHexColor(10, 10, [])).to.be.equal(undefined);

});

it('should return undefined for (10, [], 10)', function(){

expect(rgbToHexColor(10, [], 10)).to.be.equal(undefined);

});

it('should return undefined for ([], 10, 10)', function(){

expect(rgbToHexColor([], 10, 10)).to.be.equal(undefined);

});

it('should return undefined for (10, 10, {})', function(){

expect(rgbToHexColor(10, 10, {})).to.be.equal(undefined);

});

it('should return undefined for (10, {}, 10)', function(){

expect(rgbToHexColor(10, {}, 10)).to.be.equal(undefined);

});

it('should return undefined for ({}, 10, 10)', function(){

expect(rgbToHexColor({}, 10, 10)).to.be.equal(undefined);

});

});

});

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | |  | | 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); | |  | } | |  |  |   let expect = require('chai').expect; |
|  | let rgbToHexColor = require("../03. RGB to Hex").rgbToHexColor; |
|  |  |
|  | describe("rgbToHexColor(red, green, blue)", function () { |
|  | describe("Nominal cases(valid input)", function () { |
|  | it("should return #FF9EAA for (255, 158, 170)", function () { |
|  | expect(rgbToHexColor(255, 158, 170)).to.be.equal("#FF9EAA"); |
|  | }); |
|  | it("should return #0C0D0E for (12, 13, 14)", function () { |
|  | expect(rgbToHexColor(12, 13, 14)).to.be.equal("#0C0D0E"); |
|  | }); |
|  | it("should return #000000 for (0, 0, 0)", function () { |
|  | expect(rgbToHexColor(0, 0, 0)).to.be.equal("#000000"); |
|  | }); |
|  | it("should return #FFFFFF for (255, 255, 255)", function () { |
|  | expect(rgbToHexColor(255, 255, 255)).to.be.equal("#FFFFFF"); |
|  | }); |
|  | }); |
|  |  |
|  | describe("Special cases(invalid input", function () { |
|  | it("should return undefined for (-1,0,0)", function () { |
|  | expect(rgbToHexColor(-1, 0, 0)).to.be.equal(undefined); |
|  | }); |
|  | it("should return undefined for (0,-1,0)", function () { |
|  | expect(rgbToHexColor(0, -1, 0)).to.be.equal(undefined); |
|  | }); |
|  | it("should return undefined for (0,0,-1)", function () { |
|  | expect(rgbToHexColor(0, 0, -1)).to.be.equal(undefined); |
|  | }); |
|  | it("should return undefined for (256,0,0)", function () { |
|  | expect(rgbToHexColor(256, 0, 0)).to.be.equal(undefined); |
|  | }); |
|  | it("should return undefined for (0,256,0)", function () { |
|  | expect(rgbToHexColor(0, 256, 0)).to.be.equal(undefined); |
|  | }); |
|  | it("should return undefined for (0,0,256)", function () { |
|  | expect(rgbToHexColor(0, 0, 256)).to.be.equal(undefined); |
|  | }); |
|  | it("should return undefined for (3.14,0,0)", function () { |
|  | expect(rgbToHexColor(3.14, 0, 0)).to.be.equal(undefined); |
|  | }); |
|  | it("should return undefined for (0,3.14,0)", function () { |
|  | expect(rgbToHexColor(0, 3.14, 0)).to.be.equal(undefined); |
|  | }); |
|  | it("should return undefined for (0,0,3.14)", function () { |
|  | expect(rgbToHexColor(0, 0, 3.14)).to.be.equal(undefined); |
|  | }); |
|  | it('should return undefined for ("5", [3], {8:9})', function () { |
|  | expect(rgbToHexColor("5", [3], {8:9})).to.be.equal(undefined); |
|  | }); |
|  | it("should return undefined for empty input", function () { |
|  | expect(rgbToHexColor()).to.be.equal(undefined); |
|  | }); |
|  | }); |
|  | }); |

|  |
| --- |
| 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); |
|  | } |
|  |  |
|  | module.exports = { |
|  | rgbToHexColor |
|  | }; |

|  |
| --- |
| let expect = require("chai").expect; |
|  | let rgbToHexColor = require("../6\_RGBtoHex").rgbToHexColor; |
|  |  |
|  | describe("rgbToHexColor(red,green,blue)", function () { |
|  | it("should return undefined for incorrect value (shit)", function () { |
|  | expect(rgbToHexColor("shit")).to.be.equal(undefined); |
|  | }); |
|  | it("should return undefined for incorrect value (asd, 1 ,1)", function () { |
|  | expect(rgbToHexColor("asd", 1, 2)).to.be.equal(undefined); |
|  | }); |
|  | it("should return undefined for incorrect value (1, 1 ,1)", function () { |
|  | expect(rgbToHexColor("1", 1, 2)).to.be.equal(undefined); |
|  | }); |
|  | it("should return undefined for incorrect value (-1, 1 ,2)", function () { |
|  | expect(rgbToHexColor(-1, 1, 2)).to.be.equal(undefined); |
|  | }); |
|  | it("should return undefined for incorrect value (-900, 1 ,1)", function () { |
|  | expect(rgbToHexColor(-900, 1, 2)).to.be.equal(undefined); |
|  | }); |
|  | it("should return undefined for incorrect value (10, 1 ,-2)", function () { |
|  | expect(rgbToHexColor(10, 1, -2)).to.be.equal(undefined); |
|  | }); |
|  | it("should return undefined for incorrect value (2, 300 ,1)", function () { |
|  | expect(rgbToHexColor(2, 300, 2)).to.be.equal(undefined); |
|  | }); |
|  |  |
|  | it("should return #000000 for value (0, 0 ,0)", function () { |
|  | expect(rgbToHexColor(0, 0, 0)).to.be.equal("#000000"); |
|  | }); |
|  | it("should return #FF9EAA for value (255, 158 ,170)", function () { |
|  | expect(rgbToHexColor(255, 158, 170)).to.be.equal("#FF9EAA"); |
|  | }); |
|  | it("should return #0C0D0E for value (12, 13, 14)", function () { |
|  | expect(rgbToHexColor(12, 13, 14)).to.be.equal("#0C0D0E"); |
|  | }); |
|  | it("should return #FFFFFF for value (255, 255, 255)", function () { |
|  | expect(rgbToHexColor(255, 255, 255)).to.be.equal("#FFFFFF"); |
|  | }); |
|  | }) |

## 7.Add / Subtract

function createCalculator() {

let value = 0;

return {

add: function(num) { value += Number(num); },

subtract: function(num) { value -= Number(num); },

get: function() { return value; }

}

}

module.exports = {createCalculator};

let expect = require('chai').expect;

let createCalculator = require('../07AddSubtract').createCalculator;

//In Judge must be paste without this above

describe('createCalculator() - adds and substracts numbers', function(){

let calculator;

beforeEach(function(){

calculator = createCalculator();

});

it('should return 0 for get()', function(){

let value = calculator.get();

expect(value).to.be.equal(0);

});

it('should return 7 for add(7)', function(){

calculator.add(7);

let value = calculator.get();

expect(value).to.be.equal(7);

});

it('should return 17 for add(7), add(9) and add(1)', function(){

calculator.add(7);

calculator.add(9);

calculator.add(1);

let value = calculator.get();

expect(value).to.be.equal(17);

});

it('should return 8 for add(16), add(-9) and add(1)', function(){

calculator.add(16);

calculator.add(-9);

calculator.add(1);

let value = calculator.get();

expect(value).to.be.equal(8);

});

it('should return 12.8 for add(1.7), add(10.1) and add(1)', function(){

calculator.add(1.7);

calculator.add(10.1);

calculator.add(1);

let value = calculator.get();

expect(value).to.be.closeTo(12.8, 0.1);

});

it('should return -7 for subtract(7)', function(){

calculator.subtract(7);

let value = calculator.get();

expect(value).to.be.equal(-7);

});

it('should return -17 for subtract(7), subtract(9) and subtract(1)', function(){

calculator.subtract(7);

calculator.subtract(9);

calculator.subtract(1);

let value = calculator.get();

expect(value).to.be.equal(-17);

});

it('should return -8 for subtract(16), subtract(-9) and subtract(1)', function(){

calculator.subtract(16);

calculator.subtract(-9);

calculator.subtract(1);

let value = calculator.get();

expect(value).to.be.equal(-8);

});

it('should return -1.7 for subtract(0.7), subtract(0.5) and subtract(0.5)', function(){

calculator.subtract(0.7);

calculator.subtract(0.5);

calculator.subtract(0.5);

let value = calculator.get();

expect(value).to.be.closeTo(-1.7, 0.1);

});

it('should return 8 for add(7), subtract(9) and add(10)', function(){

calculator.add(7);

calculator.subtract(9);

calculator.add(10);

let value = calculator.get();

expect(value).to.be.equal(8);

});

it('should return 0 for add(0), subtract(0) and add(0)', function(){

calculator.add(0);

calculator.subtract(0);

calculator.add(0);

let value = calculator.get();

expect(value).to.be.equal(0);

});

it('should return 1 for add(1), subtract(0) and add(0)', function(){

calculator.add(1);

calculator.subtract(0);

calculator.add(0);

let value = calculator.get();

expect(value).to.be.equal(1);

});

it('should return 8 for add("10"), subtract("8") and add("6")', function(){

calculator.add("10");

calculator.subtract("8");

calculator.add("6");

let value = calculator.get();

expect(value).to.be.equal(8);

});

it('should return 8 for add(10), subtract("8") and add("6")', function(){

calculator.add(10);

calculator.subtract("8");

calculator.add("6");

let value = calculator.get();

expect(value).to.be.equal(8);

});

it('should return NaN for add(string)', function(){

calculator.add('hi');

let value = calculator.get();

expect(value).to.be.NaN;

});

it('should return NaN for subtract(string)', function(){

calculator.subtract('hi');

let value = calculator.get();

expect(value).to.be.NaN;

});

});

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | | function createCalculator() { | |  | let value = 0; | |  | return { | |  | add: function(num) { value += Number(num); }, | |  | subtract: function(num) { value -= Number(num); }, | |  | get: function() { return value; } | |  | } | |  | } | |  |  | |  | module.exports = { createCalculator } |   let expect = require('chai').expect; |
|  | let createCalculator = require("../04. Add Subtract").createCalculator; |
|  |  |
|  | describe("createCalculator()", function () { |
|  | let calc; |
|  |  |
|  | beforeEach(function () { |
|  | calc = createCalculator(); |
|  | }); |
|  |  |
|  | it("should return 0 for get;", function () { |
|  | let value = calc.get(); |
|  | expect(value).to.be.equal(0); |
|  | }); |
|  | it("should return 5 after add(2); add(3);", function () { |
|  | calc.add(2); |
|  | calc.add(3); |
|  | let value = calc.get(); |
|  | expect(value).to.be.equal(5); |
|  | }); |
|  | it("shoul return -5 after subtract(3); subtract(2)", function () { |
|  | calc.subtract(3); |
|  | calc.subtract(2); |
|  | let value = calc.get(); |
|  | expect(value).to.be.equal(-5); |
|  | }); |
|  | it("should return 4.2 after add(5.3); subtract(1.1);", function () { |
|  | calc.add(5.3); |
|  | calc.subtract(1.1); |
|  | let value = calc.get(); |
|  | expect(value).to.be.equal(5.3 - 1.1); |
|  | }); |
|  | it("should return 2 after add(10); subtract('7'); add('-2'); subtract(-1)", function () { |
|  | calc.add(10); |
|  | calc.subtract('7'); |
|  | calc.add('-2'); |
|  | calc.subtract(-1); |
|  | let value = calc.get(); |
|  | expect(value).to.be.equal(2); |
|  | }); |
|  | it("should return NaN for add(string)", function () { |
|  | calc.add('hello'); |
|  | let value = calc.get(); |
|  | expect(value).to.be.NaN; |
|  | }); |
|  | it("should return NanN for subtarct(string)", function () { |
|  | calc.subtract('hello'); |
|  | let value = calc.get(); |
|  | expect(value).to.be.NaN; |
|  | }); |
|  | }); |

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

|  |
| --- |
| let expect = require("chai").expect; |
|  | let createCalculator = require("../7\_AddSubtract").createCalculator; |
|  |  |
|  | describe("createCalculator", function () { |
|  | let calc = createCalculator(); |
|  |  |
|  | beforeEach(function () { |
|  | calc = createCalculator(); |
|  | }) |
|  |  |
|  | it("should return 5 after {add 3; add 2}", function () { |
|  | calc.add(3); |
|  | calc.add(2); |
|  | let value = calc.get(); |
|  | expect(value).to.be.equal(5); |
|  | }) |
|  | it("should return 0 after {get}", function () { |
|  | let value = calc.get(); |
|  | expect(value).to.be.equal(0); |
|  | }) |
|  | it("should return -5 after {subtract(3);subtract(2)}", function () { |
|  | calc.subtract(3); |
|  | calc.subtract(2); |
|  | let value = calc.get(); |
|  | expect(value).to.be.equal(-5); |
|  | }) |
|  | it("should return 4.2 after {add(5.3);subtract(1.1)}", function () { |
|  | calc.add(5.3); |
|  | calc.subtract(1.1); |
|  | let value = calc.get(); |
|  | expect(value).to.be.equal(5.3 - 1.1); |
|  | }) |
|  | it("should return NaN after {add('hello')}", function () { |
|  | calc.add('hello'); |
|  | let value = calc.get(); |
|  | expect(value + "").to.be.equal(NaN + ""); |
|  | }) |
|  | it("should return NaN after {subtract('hello')}", function () { |
|  | calc.subtract('hello'); |
|  | let value = calc.get(); |
|  | expect(value + "").to.be.equal(NaN + ""); |
|  | }) |
|  | }) |

|  |
| --- |
| { |
|  | // Use IntelliSense to learn about possible attributes. |
|  | // Hover to view descriptions of existing attributes. |
|  | // For more information, visit: https://go.microsoft.com/fwlink/?linkid=830387 |
|  | "version": "0.2.0", |
|  | "configurations": [ |
|  | { |
|  | "type": "node", |
|  | "request": "launch", |
|  | "name": "Launch Program", |
|  | "program": "${workspaceFolder}/tests\\4\_SumOfNumbers-mocha-tests.js" |
|  | } |
|  | ] |
|  | } |

{

// Use IntelliSense to learn about possible attributes.

// Hover to view descriptions of existing attributes.

// For more information, visit: https://go.microsoft.com/fwlink/?linkid=830387

"version": "0.2.0",

"configurations": [

{

"type": "node",

"request": "launch",

"name": "Mocha Tests",

"program": "${workspaceFolder}/node\_modules/mocha/bin/\_mocha",

"args": [

"-u",

"tdd",

"--timeout",

"999999",

"--colors",

"${workspaceFolder}/{,!(node\_modules)/}\*/\*/\*-tests.js"

],

"internalConsoleOptions": "openOnSessionStart"

},

{

"type": "node",

"request": "launch",

"name": "Launch Program",

"program": "${workspaceFolder}/JSCoreAdvanced\\02ErrorHandlingAndUnitTestingWithMochaLab\\test\\07AddSubtract-tests.js"

}

]

}

Working solution for running tests:

{

// Use IntelliSense to learn about possible attributes.

// Hover to view descriptions of existing attributes.

// For more information, visit: https://go.microsoft.com/fwlink/?linkid=830387

"version": "0.2.0",

"configurations": [

{

"type": "node",

"request": "launch",

"name": "Mocha Tests",

"program": "${workspaceFolder}/node\_modules/mocha/bin/\_mocha",

"args": [

"-u",

"tdd",

"--timeout",

"999999",

"--colors",

"${workspaceFolder}//JSCoreAdvanced\\02ErrorHandlingAndUnitTestingWithMochaLab\\test\\06RGBtoHex-tests.js"//It must be done for each file

],

"internalConsoleOptions": "openOnSessionStart"

},

{

"type": "node",

"request": "launch",

"name": "Launch Program",

"program": "${workspaceFolder}/JSCoreAdvanced\\02ErrorHandlingAndUnitTestingWithMochaLab\\test\\07AddSubtract-tests.js"

}

]

}