Unit Testing

Error and Exception Handling, Unit Testing, Test Cases, Assertions

SoftUni Team

Technical Trainers

Software University

http://softuni.bg

Table of Contents

Have a Question?

SoftUni Foundation

sli.do

#JS-CORE

Error Handling

Concepts, Examples, Exceptions

Types of Errors

There are **three types** of errors in programming:

- **Syntax Errors** - occur at compile time.
- **Runtime Errors** - occur during execution (after compilation).
- **Logical Errors** - occur when you make a mistake in the logic that drives your script and you do not get the result you expected.

Error Handling - Concepts

A function failed to do what its name suggests should:

- Return a special value (e.g. **undefined** / **false** / **-1**).
- Throw an **exception** / **error**.

```
let str = "Hello, SoftUni";
console.log(str.indexOf("Sofia")); // -1
// Special case returns a special value to indicate "not found".
```

Error Handling

The fundamental **principle** of error handling says that a function (method) should either:

- Do what its **name** suggest
- Indicate a **problem**
- Any other behavior is **incorrect**

Error Handling - Exceptions (Errors)

SoftUni Foundation

Exception - a function is unable to do its work (**fatal error**).

```
let arr = new Array(-1);   // Uncaught RangeError
```

```
let bigArr = new Array(9999999999); // RangeError
```

```
let index = undefined.indexOf("hi"); // TypeError
```

```
console.log(asfd);      // Uncaught ReferenceError
```

```
console.print('hi');          // Uncaught TypeError
```

Error Handling - Special Values

```
let sqrt = Math.sqrt(-1); // NaN (special value)
```

```
let sub = "hello".substring(2, 1000); // llo
let sub = "hello".substring(-100, 100); // hello
// Soft error - substring still does its job: takes all
available chars
```

```
let inv = new Date("Christmas"); // Invalid Date
let dt = inv.getDate(); // NaN
```

Unexpected (Not Obvious) Behavior

In JavaScript, the **first** month (January) is month number **0**, so December **returns** month **number 11**.

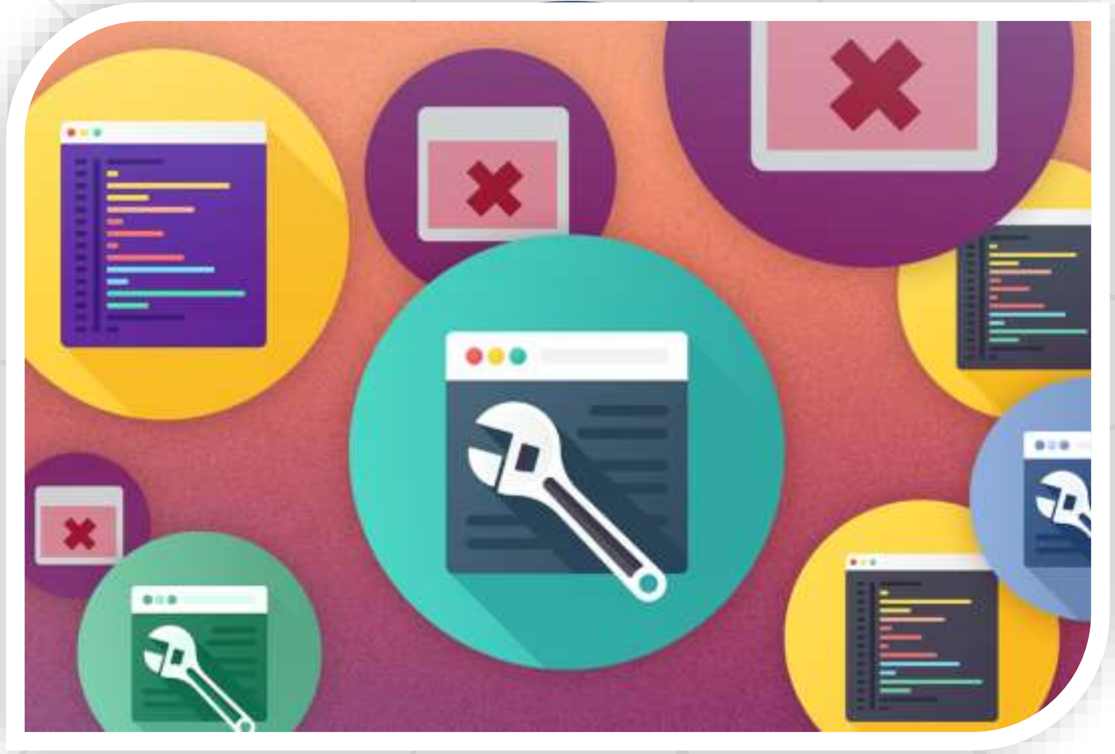
```
let date = new Date(2016, 1, 20);    // Feb 20 2016
```

```
let date1 = new Date(1, 1, 1);       // Feb 01 1901
```

```
let dateMinus1 = new Date(-1, -1, -1); // Nov 29 -2
```

```
let dateNext = new Date(2016, 1, 30)   // Mar 01 2016 (next month)
```

```
let datePrev = new Date(2016, -1, 30); // Dec 30 2015 (prev month)
```



Exception Handling

Throwing / Catching Errors

Throwing Errors (Exceptions)

The **throw** statement lets you create custom errors

- **General Error**
  - throw new Error("**Invalid state**")
- **Range Error**
  - throw new RangeError("**Invalid index**")
- **Type Error**
  - throw new TypeError("**String expected**")
- **Reference Error**
  - throw new ReferenceError("**Missing age**")

Try - catch

SoftUni Foundation

- The **try** statement lets you test a block of code for errors.
- The **catch** statement lets you handle the error.
- **Try** and **catch** come in pairs:

```
try {
  // Code that can throw an exception
  // Some other code → not executed in case of error!
} catch (ex) {
  // This code is executed in case of exception
  // Ex holds the info about the exception
}
```

Exception Properties

- An **Error object** with properties will be created.

```
try {
  throw new RangeError("Invalid range.");
  console.log("This will not be executed.");
} catch (ex) {
  console.log("Exception object: " + ex);
  console.log("Type: " + ex.name);
  console.log("Message: " + ex.message);
  console.log("Stack: " + ex.stack);
}
```

Unit Testing

Definition, Structure, Examples, Frameworks

Unit Testing

A **unit test** is a piece of code that checks whether a piece of functionality is working as expected.

- Allow developers to see **where** (and **why**) **errors occur**.

```
function sortNums(arr) {
    arr.sort((a,b) => a - b);
}
```

```
let nums = [2, 15, -2, 4];
sortNums(nums);

if (JSON.stringify(nums) === "[-2,2,4,15]") {
    console.error("They are equal!");
}
```

Unit Tests Structure

The **AAA** Pattern: **Arrange**, **Act**, **Assert**

```
// Arrange all necessary preconditions and inputs
let nums = [2, 15, -2, 4];
// Act on the object or method under test
sortNums(nums);
// Assert that the obtained results are what we expect
if (JSON.stringify(nums) === "[-2,2,4,15]"){
    console.error("They are equal!");
}
```

Unit Testing Frameworks

- JS Unit Testing:
  - Mocha, QUnit, Unit.js, Jasmine
- Assertion frameworks (perform checks):
  - Chai, Assert.js, Should.js
- Mocking frameworks (mocks and stubs):
  - Sinon, JMock, Mockito, Moq

Mocha and Chai

Unit Testing with Mocha and Chai

What is Mocha?

Feature-rich JS test framework

- Provides common testing functions including **it**, **describe** and the **main function** that runs tests.

```
describe("title", function() {
  it("title", function() {…});
});
```

- Usually used together with **Chai**.

What is Chai?

Chai is a **library** with many assertions

- It allows to use a lot of different assertions such as **assert.equal**.

```
let assert = require("chai").assert;
describe("pow", function() {
   it("2 raised to power 3 is 8", function() {
    assert.equal(pow(2, 3), 8);
  });
});
```

WebStorm Configuration

Installing Mocha and Chai

- Installing Mocha and Chai through **npm**

```
npm -g install mocha
```

```
npm -g install chai
```

- Check if Mocha is installed

```
mocha --version
```

- Install via **npm** and use the **chai.js** file found within the download.

```
<script src="./node_modules/chai/chai.js"></script>
```

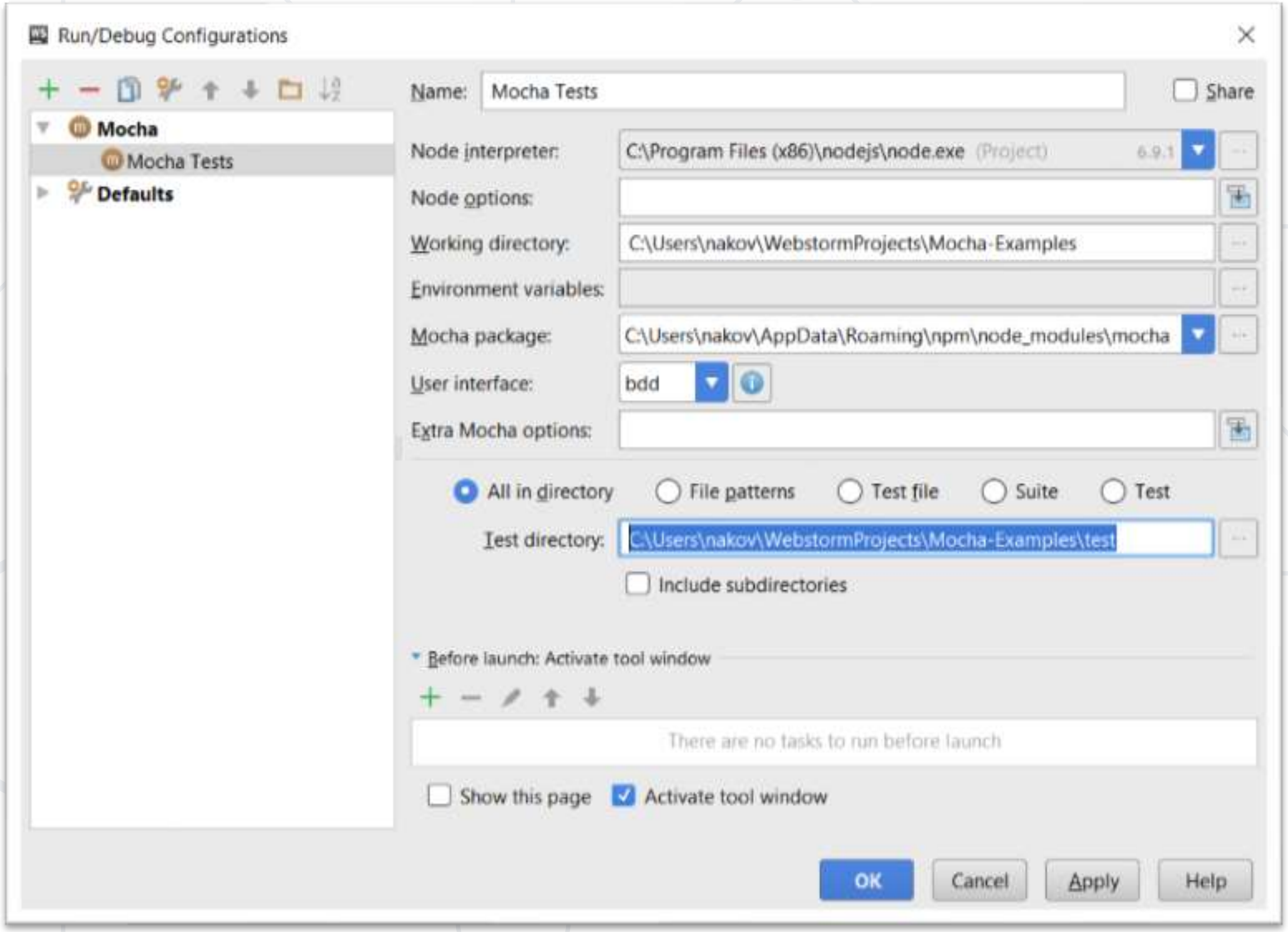
Configuring NODE_PATH

- By default Node.js does not find its globally-installed modules.
- You need to set the **NODE_PATH** environment variable.

```
rem for any future sessions
setx NODE_PATH %AppData%\npm\node_modules
rem for current session
set NODE_PATH=%AppData%\npm\node_modules
```

- You may need to restart your IDE after changing **NODE_PATH**.

Configuring Mocha in WebStorm


Run/Debug Configurations

Mocha
Mocha Tests
Defaults

Name: Mocha Tests
[ ] Share

Node interpreter: C:\Program Files (x86)\nodejs\node.exe (Project) 6.9.1
Node options:
Working directory: C:\Users\nakov\WebstormProjects\Mocha-Examples
Environment variables:
Mocha package: C:\Users\nakov\AppData\Roaming\npm\node_modules\mocha
User interface: bdd
Extra Mocha options:

[✓ All in directory] File patterns Test file Suite Test

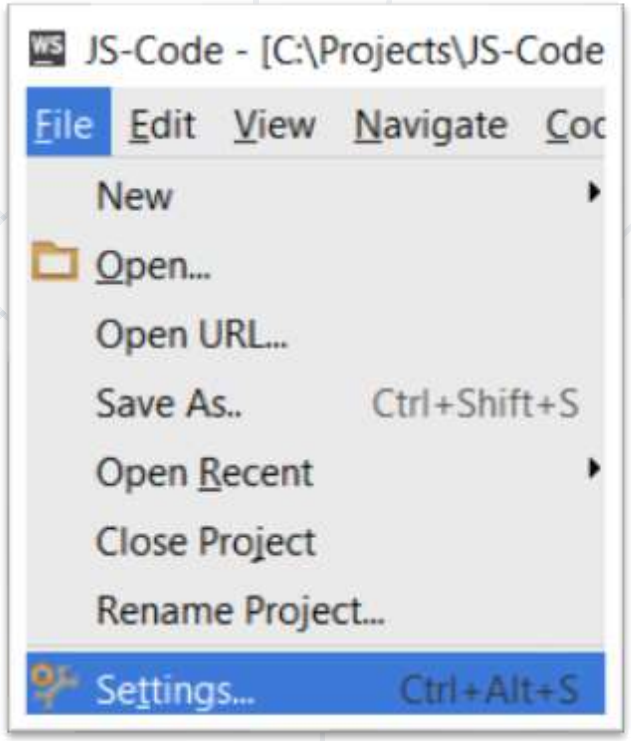
Test directory: C:\Users\nakov\WebstormProjects\Mocha-Examples\test
[ ] Include subdirectories

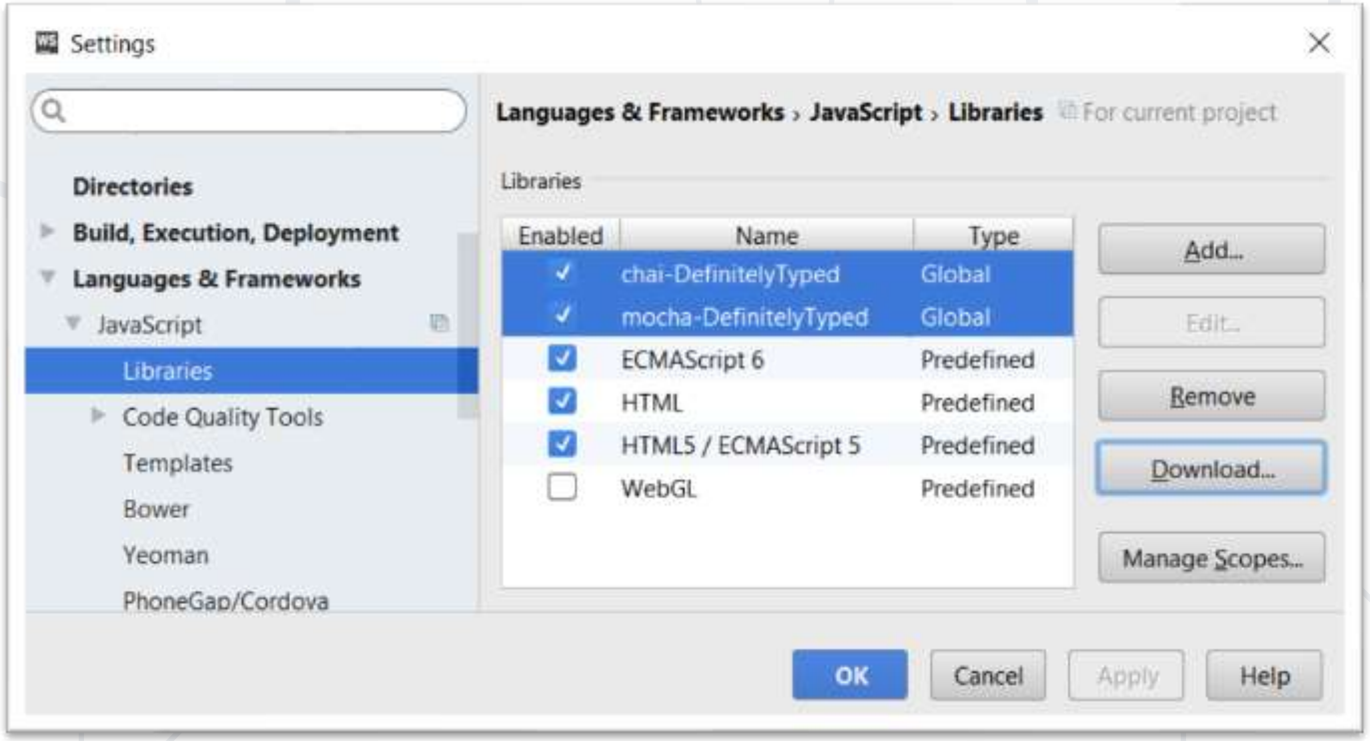
Before launch: Activate tool window
There are no tasks to run before launch

[ ] Show this page
[✓] Activate tool window

Configuring Libraries in WebStorm

- To get the "**auto complete**" and "**syntax checks**" working for Mocha and Chai, add them as **libraries** in WebStorm.



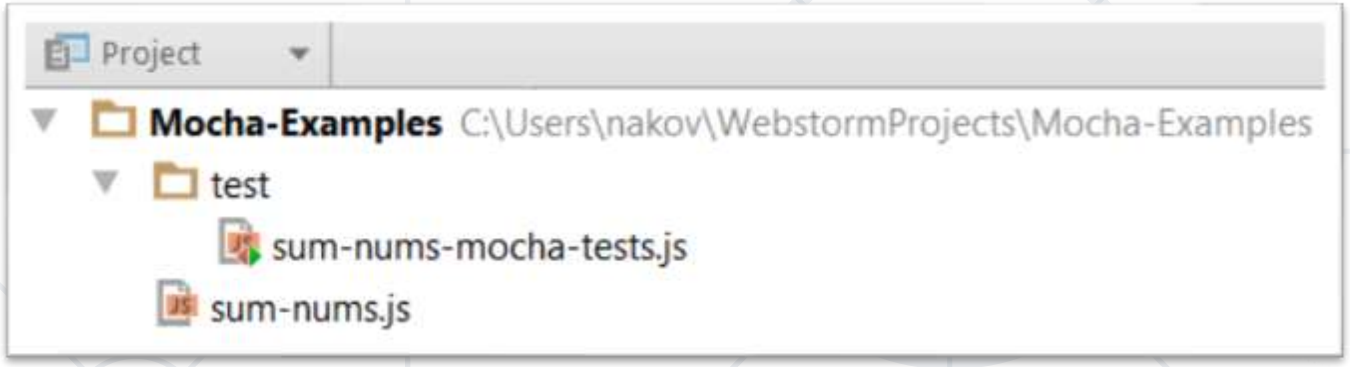


Libraries

Enabled	Name	Type
✓	chai-DefinitelyTyped	Global
✓	mocha-DefinitelyTyped	Global
✓	ECMAScript 6	Predefined
✓	HTML	Predefined
✓	HTML5 / ECMAScript 5	Predefined
	WebGL	Predefined

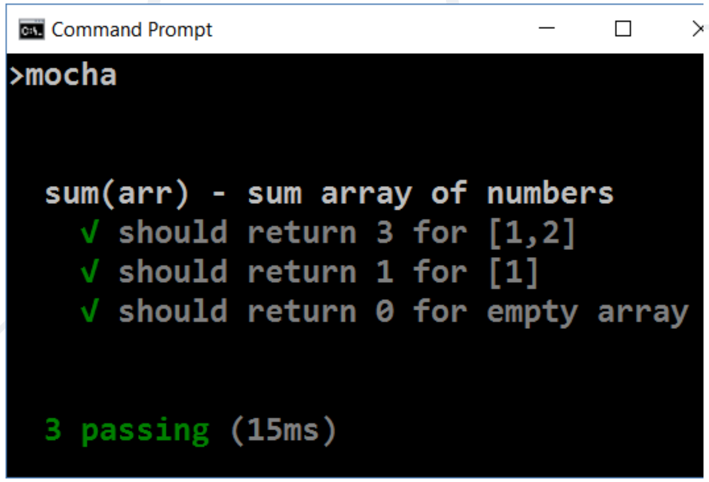
Running Mocha

1. Create folder **"test"** in your JS project.



```
Mocha-Examples C:\Users\nakov\WebstormProjects\Mocha-Examples
  test
    sum-nums-mocha-tests.js
  sum-nums.js
```

2. Put your test code in **test/{test-group-name}.js**
3. Run **mocha** from the console



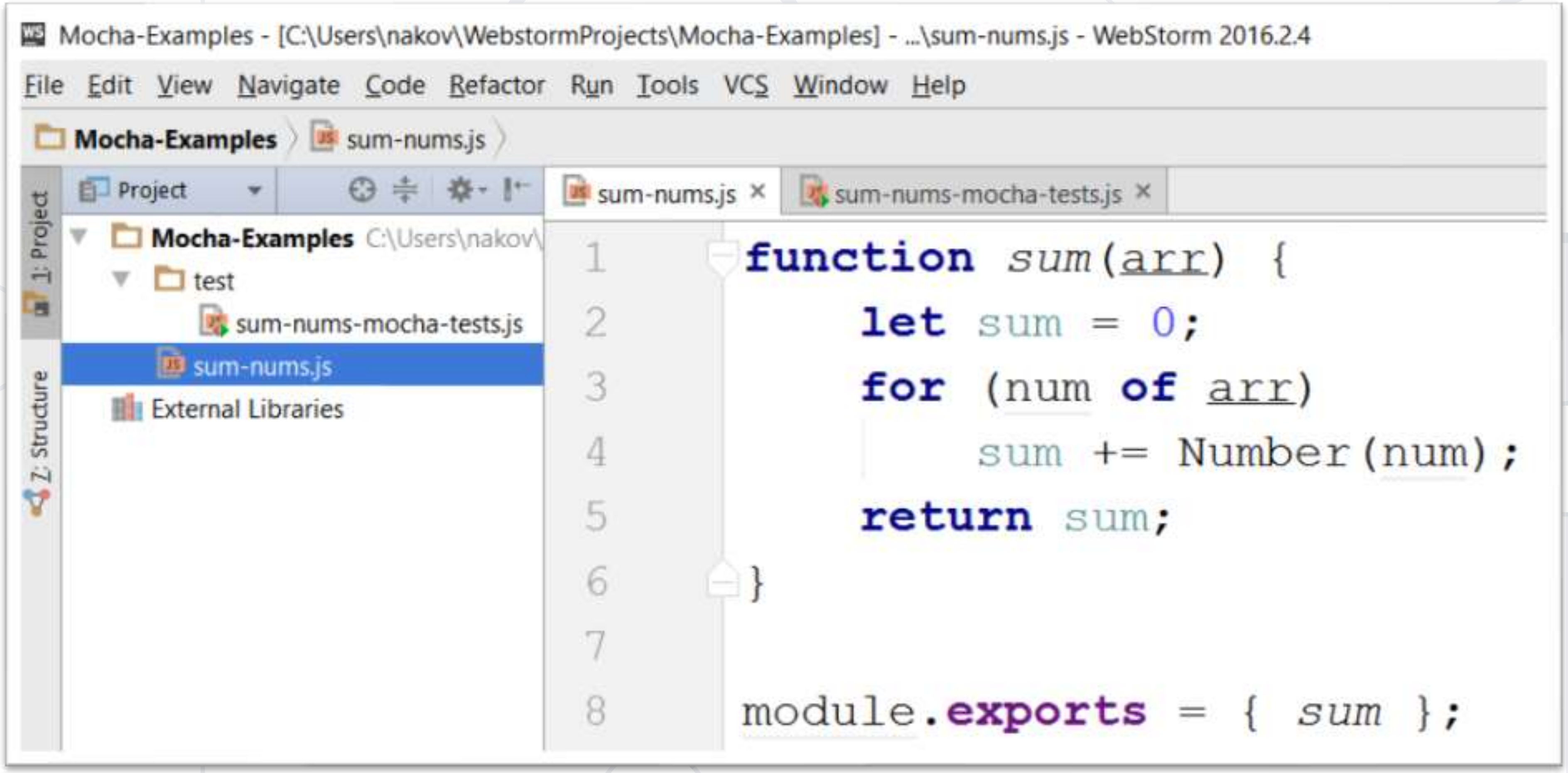
```
>mocha


  sum(arr) - sum array of numbers
    √ should return 3 for [1,2]
    √ should return 1 for [1]
    √ should return 0 for empty array


  3 passing (15ms)
```

Sample Project

- Source code to be tested



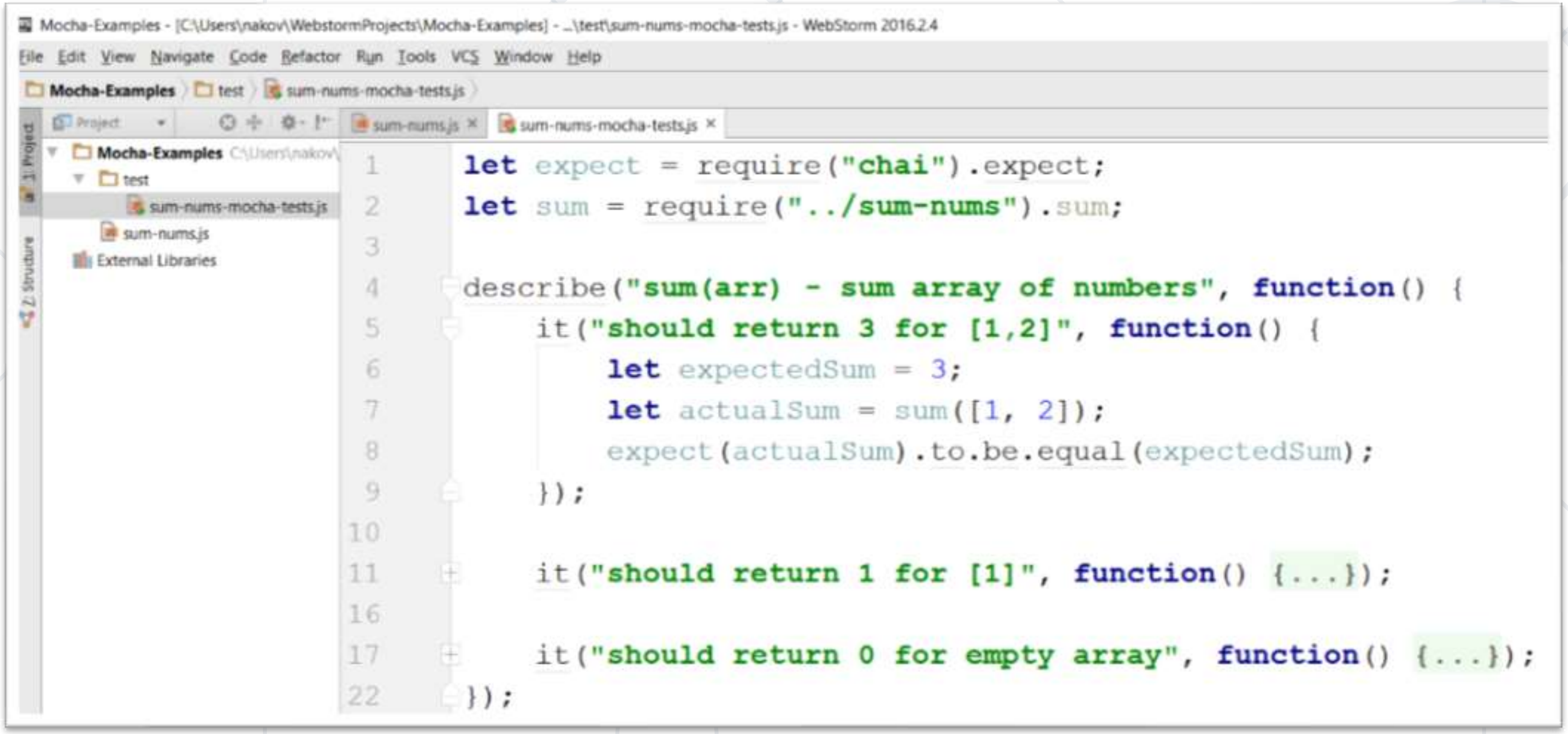
Mocha-Examples - [C:\Users\nakov\WebstormProjects\Mocha-Examples] - ...\sum-nums.js - WebStorm 2016.2.4
File Edit View Navigate Code Refactor Run Tools VCS Window Help
Mocha-Examples
sum-nums.js
Project
Mocha-Examples C:\Users\nakov\
test
sum-nums-mocha-tests.js
sum-nums.js
External Libraries
1: Project
Z: Structure
sum-nums.js
sum-nums-mocha-tests.js

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

module.exports = { sum };
```

Sample Unit Tests

- Tests for the **sum(arr)** function.


Mocha-Examples - [C:\Users\nakov\WebstormProjects\Mocha-Examples] - ...\test\sum-nums-mocha-tests.js - WebStorm 2016.2.4
File Edit View Navigate Code Refactor Run Tools VCS Window Help
Mocha-Examples test sum-nums-mocha-tests.js
Project
sum-nums.js
sum-nums-mocha-tests.js
Mocha-Examples C:\Users\nakov\
test
sum-nums-mocha-tests.js
sum-nums.js
External Libraries
1: Project
Z: Structure

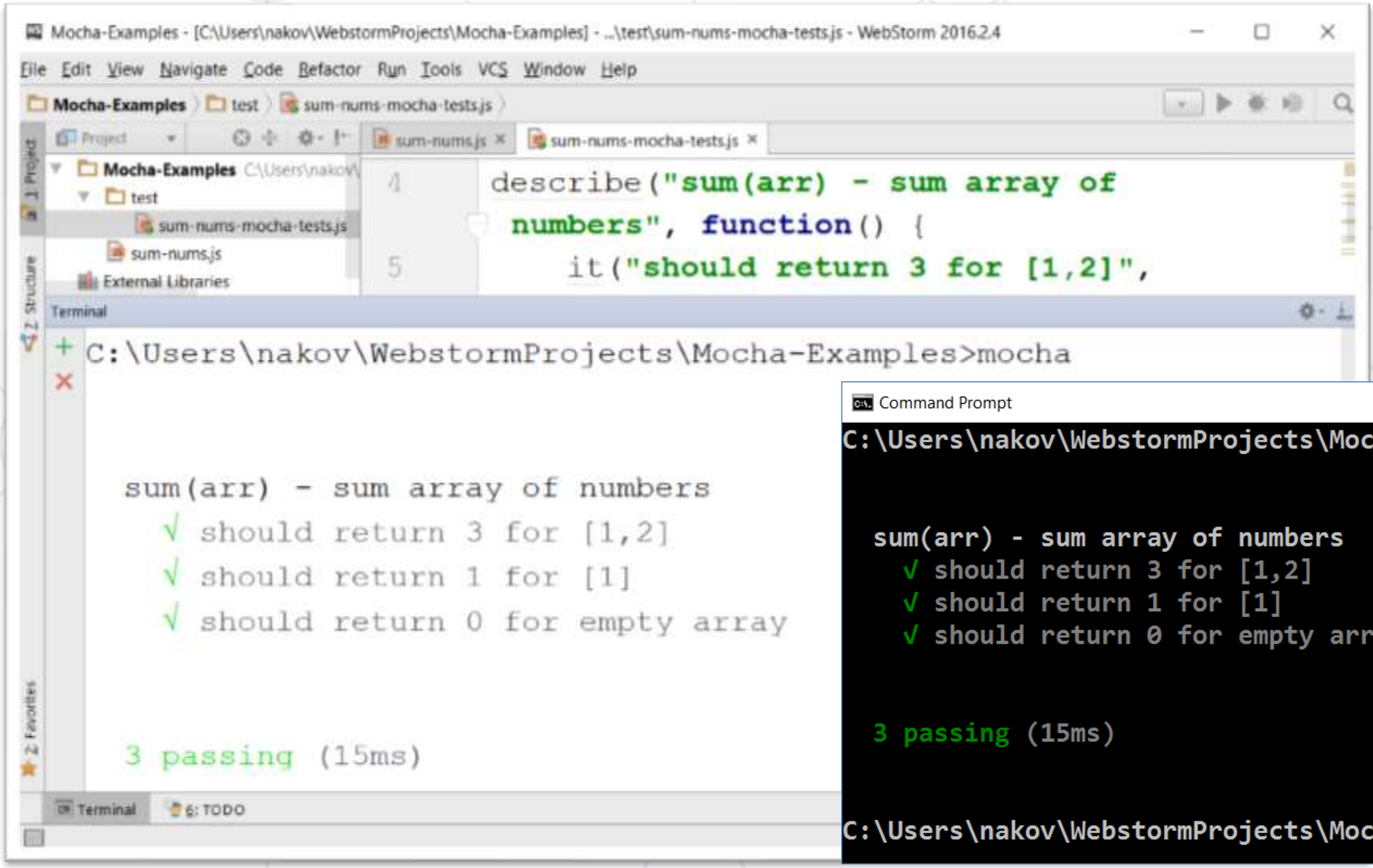
```javascript
let expect = require("chai").expect;
let sum = require("../sum-nums").sum;

describe("sum(arr) - sum array of numbers", function() {
    it("should return 3 for [1,2]", function() {
        let expectedSum = 3;
        let actualSum = sum([1, 2]);
        expect(actualSum).to.be.equal(expectedSum);
    });

    it("should return 1 for [1]", function() {...});

    it("should return 0 for empty array", function() {...});
});
```

Running the Tests

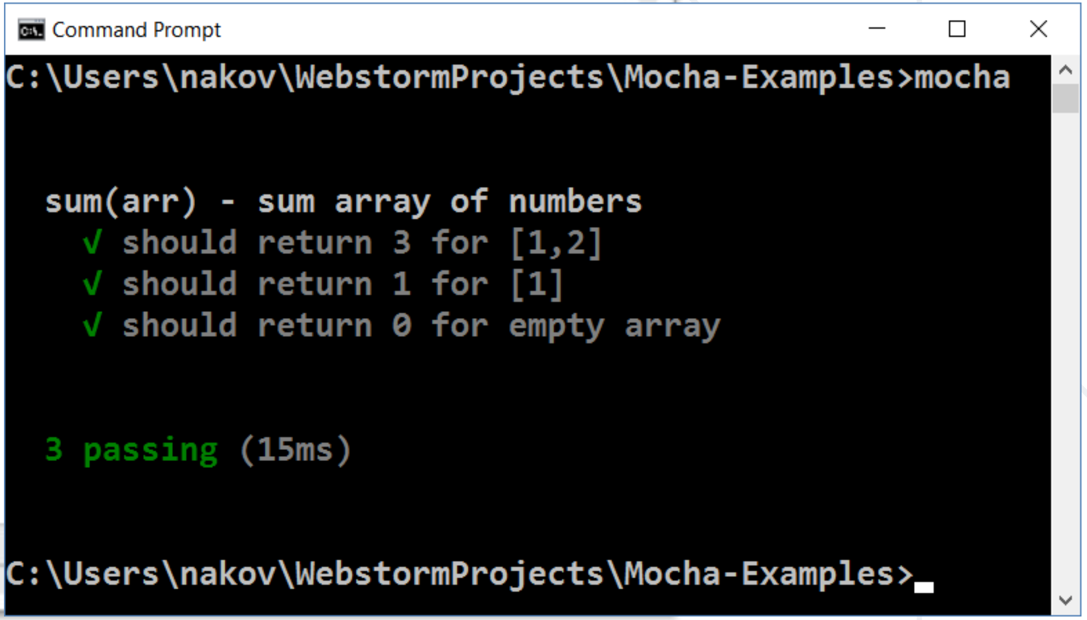

Mocha-Examples - [C:\Users\nakov\WebstormProjects\Mocha-Examples] - ...\test\sum-nums-mocha-tests.js - WebStorm 2016.2.4
File Edit View Navigate Code Refactor Run Tools VCS Window Help
Mocha-Examples test sum-nums-mocha-tests.js
Project
sum-nums.js
sum-nums-mocha-tests.js
Mocha-Examples C:\Users\nakov\
test
sum-nums-mocha-tests.js
sum-nums.js
External Libraries
Terminal
Terminal
6: TODO

```
describe("sum(arr) - sum array of
 numbers", function() {
   it("should return 3 for [1,2]",
```

```
C:\Users\nakov\WebstormProjects\Mocha-Examples>mocha


  sum(arr) - sum array of numbers
    √ should return 3 for [1,2]
    √ should return 1 for [1]
    √ should return 0 for empty array


  3 passing (15ms)
```



```
C:\Users\nakov\WebstormProjects\Mocha-Examples>mocha


  sum(arr) - sum array of numbers
    √ should return 3 for [1,2]
    √ should return 1 for [1]
    √ should return 0 for empty array


  3 passing (15ms)


C:\Users\nakov\WebstormProjects\Mocha-Examples>
```

VS Code Configuration

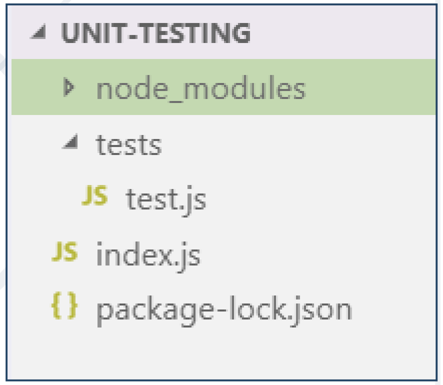
NPM Install

- Install **Mocha**:

```
npm install mocha
```

- Install **Chai**:

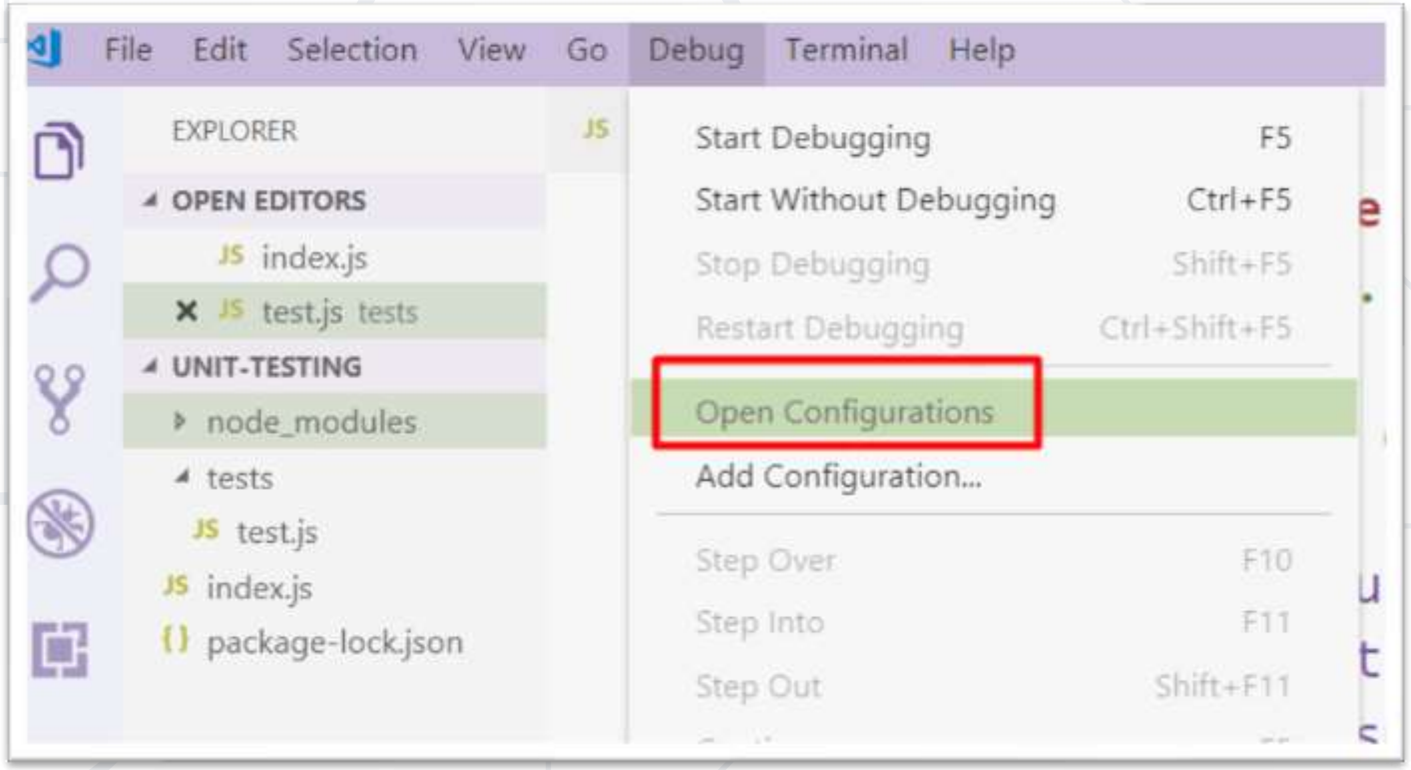
```
npm install chai
```



UNIT-TESTING
- node_modules
- tests
  - test.js
- index.js
- package-lock.json

Configuration - launch.json

1. Open your **launch.json** file:



Configuration - launch.json

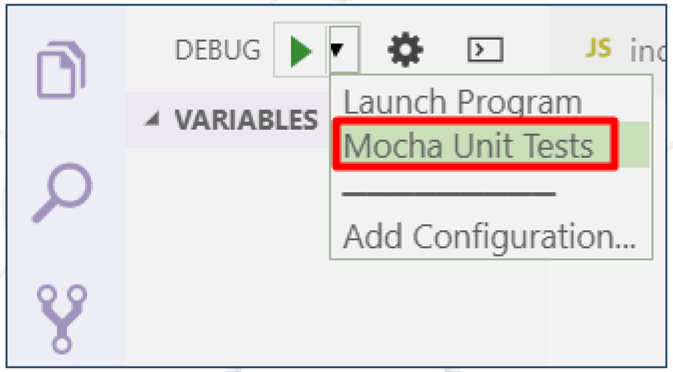
2. Add the following code:

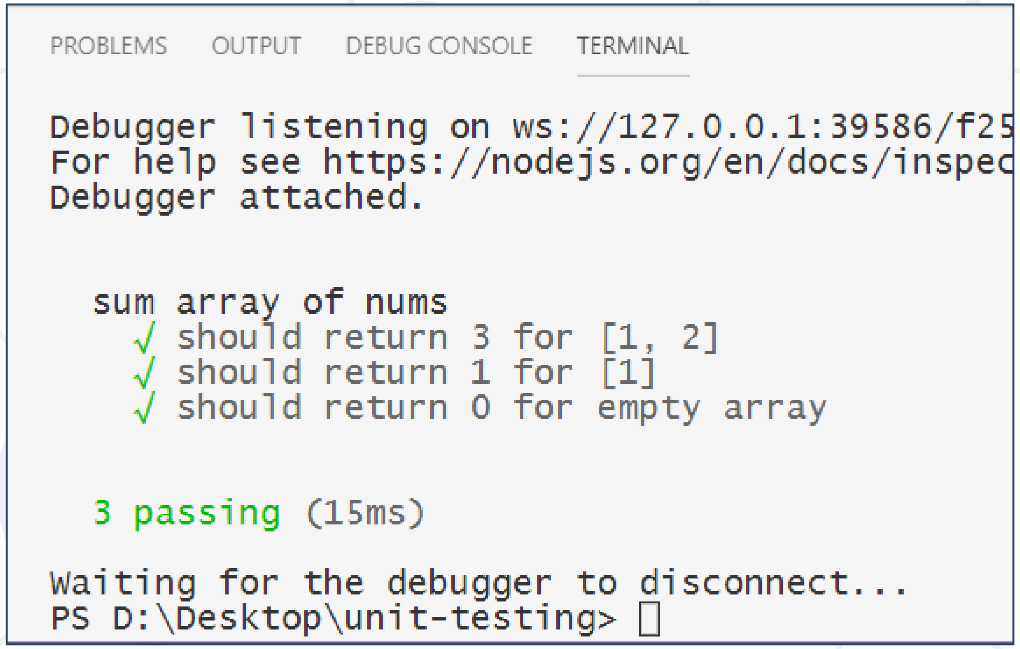
```
{
"type": "node",
"request": "launch",
"name": "Mocha Unit Tests",
"program":
"${workspaceFolder}node_modules\\.bin\\_mocha",
"runtimeArgs": [
"${workspaceFolder}/tests/test.js"
],
"console": "externalTerminal"
}
```

Path to your tests file

Configuration - launch.json

3. Choose debugging configuration:





```
Debugger listening on ws://127.0.0.1:39586/f25
For help see https://nodejs.org/en/docs/inspec
Debugger attached.


  sum array of nums
    √ should return 3 for [1, 2]
    √ should return 1 for [1]
    √ should return 0 for empty array


  3 passing (15ms)

Waiting for the debugger to disconnect...
PS D:\Desktop\unit-testing> 
```

Test Groups and Test Classes

```
let expect = require("chai").expect;
```

```
describe("Test group #1", function() {
  it("should… when…", function() {
    expect(actual).to.be.equal(expected);
  });
  it("should… when…", function() { … });
});
describe("Test group #2", function() {
  it("should… when…", function() {
    expect(actual).to.be.equal(expected);
  });
  …
});
```

Live Exercises

Unit-Testing

Summary

- A **function** should do what its **name suggests**.
- **Exceptions** are thrown when a function is **unable** to do its **work**.
- The **throw** statement lets you create **custom errors**.
- **Unit Tests** check whether a piece of code **works as expected**.
- **Mocha** is a JS **test framework** that is usually used together with **Chai** (**assertion library**).

Questions?

SoftUni

Software University

SoftUni Svetlina

SoftUni Creative

SoftUni Digital

SoftUni Foundation

SoftUni Kids

SoftUni Diamond Partners

SoftUni Foundation

xssoftware

SBTech we know sports

telenor

SoftwareGroup doing it right

NETPEAK

SmartIT

Postbank Решения за твоето утре

SUPER HOSTING ®.BG

INDEAVR Serving the high achievers

INFRAGISTICS®

LIEBHERR

æternity

SoftUni Organizational Partners

SoftUni Foundation

IS ИНФОРМАЦИОННО ОБСЛУЖВАНЕ

OneBit SOFTWARE

Lukanet.com

WORLD OF MYTHS

Trainings @ Software University (SoftUni)

- Software University – High-Quality Education and Employment Opportunities
  - softuni.bg
- Software University Foundation
  - http://softuni.foundation/
- Software University @ Facebook
  - facebook.com/SoftwareUniversity
- Software University Forums
  - forum.softuni.bg

License