**Instructions**

Close

**Lab instructions**

0 of 1 completed

Writing a Unit Test Lab

​

**Introduction:**

Unit testing is a critical skill in software development, enabling you to verify that individual units of code work as expected. This lab introduces Jest, a popular JavaScript testing framework, and walks you through writing a unit test for a simple function.

**Goal:**

The goal of this lab is to familiarize learners with writing and running unit tests using Jest.

**Objectives:**

* Install Jest as a development dependency using npm.
* Configure the **package.json** file for Jest testing.
* Write and export JavaScript functions for testing.
* Write unit tests using Jest's **test()** and **expect()** functions.

**Learner Instructions:**

**Lab Tasks:**

**Task 1:** Add Jest as a Development Dependency

**Objective:** Learn how to install Jest using npm and configure it as a development dependency in a JavaScript project.

**Steps:**

**Step 1:** Navigate to **Terminal**>**New Terminal**and execute the **npm install jest --save-dev** command to install Jest.

**Step 2:** Verify the installation:

* Open the **package.json** file present under the **Jest-Testing** folder.
* Confirm the presence of the following entry:

1

2

3

"devDependencies": {

    "jest": "^29.7.0"

}

**Task 2:** Update the **test** Script

**Objective:** Understand how to modify the scripts section of package.json to include a Jest test runner.

**Steps:**

**Step 1:** In the **package.json** file, locate the "**scripts**" section and update the "**test**" entry as follows:

1

2

3

"scripts": {

    "test": "jest"

}

**Task 3:** Code the **timesTwo** Function

**Objective:** Implement a JavaScript function to perform a basic calculation and export it for use in a separate module.

**Steps:**

**Step 1:** Open the **timesTwo.js** file present under the **Jest-Testing** folder.

**Step 2:** Implement the **timesTwo** function:

* It should take a number as input and return the number multiplied by 2.

**Step 3:** Export the function as a module:

1

module.exports = timesTwo;

**Step 4:** After successfully modifying the **timesTwo.js** file, navigate to **File** > **Save** to save changes in the file.

**Task 4:** Write the First Test

**Objective:** Write a unit test using Jest's **test()** and **expect()** functions to verify the correctness of the **timesTwo** function.

**Steps:**

**Step 1:** Open the **timesTwo.test.js** file present under the **Jest-Testing** folder.

**Step 2:** Write a test for the **timesTwo** function:

* Use the **test()** function with the description: "**returns the number times 2**".
* Ensure the test checks that calling **timesTwo(10**) returns **20**.

1

2

3

test('returns the number times 2', () => {

    expect(timesTwo(10)).toBe(20);

});

**Step 3:** After successfully modifying the **timesTwo.test.js** file, navigate to **File** > **Save** to save changes in the file.

**Task 5:** Run the First Test

**Objective:** Execute Jest tests via npm, interpret the results, and ensure the function meets its expected behavior.

**Steps:**

**Step 1:** In the terminal, execute the **npm test** command to run the test script.

**Step 2:** Verify that the test passes, and observe the output in the terminal.

**Key Takeaways:**

* Unit tests help validate the behavior of small, isolated parts of a codebase.
* Jest provides an easy-to-use API for writing and running tests.
* Use **npm install --save-dev** to add packages needed during development, like Jest.

**Final Step: Submit Your Code:**

* Go to **File** > **Save** to ensure your work is saved.
* **Submit your assignment**: Click the "**Submit Assignment"** button in the Lab toolbar.
  + Your code will be **autograded** and feedback will be available shortly on the **Grades** tab.