Lesson 08 Demo 01

Testing the DOM Using Jest

Objective: To demonstrate the implementation of a DOM (document object model) testing example using Jest

Tools required: Visual Studio Code, Node.js, and npm

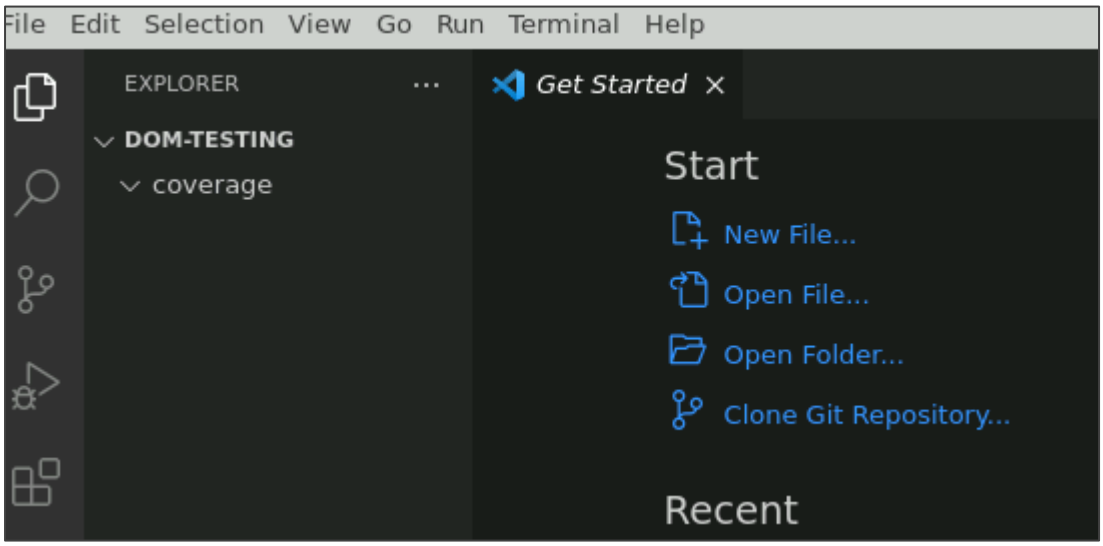
Prerequisites: Good understanding of JavaScript, HTML, DOM, and some knowledge of Jest

Steps to be followed:

1. Create a Node project
2. Create an HTML file
3. Create a JavaScript file
4. Write a Jest test
5. Run the Jest test

Step 1: Create a Node project

1.1 Open the Visual Studio Code editor and create a folder named **dom-testing**



1.2 Open the terminal and initialize a new Node.js project by running the following command and accepting the default settings:
npm init -y

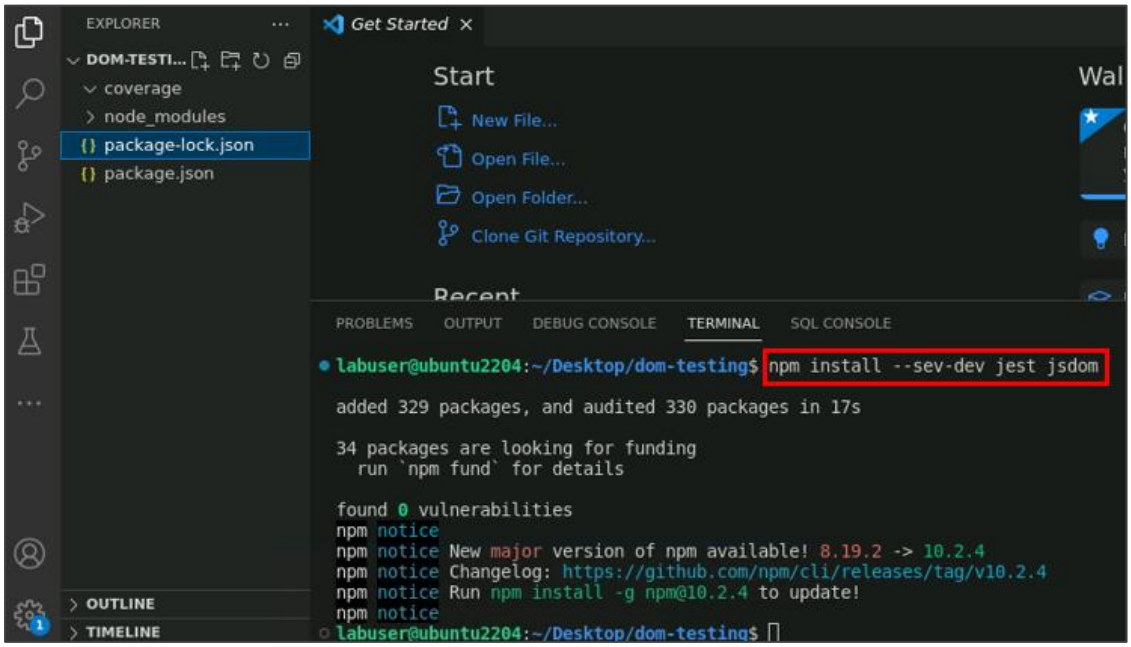
```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    SQL CONSOLE

labuser@ubuntu2204:~/Desktop/dom-testing$ npm init -y
Wrote to /home/labuser/Desktop/dom-testing/package.json:

{
  "name": "dom-testing",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC"
}

labuser@ubuntu2204:~/Desktop/dom-testing$
```

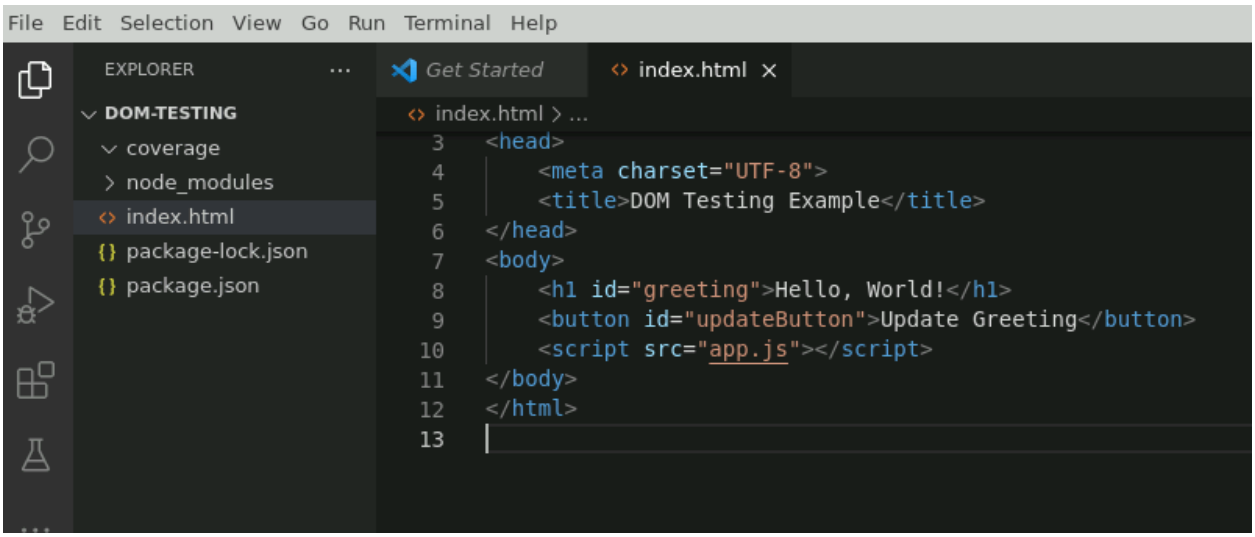
1.3 Install Jest as a development dependency along with **jsdom**, which provides an environment for testing, by running the following command:
npm install --save-dev jest jsdom



Step 2: Create an HTML file

2.1 Create a **index.html** file in the root directory of project and add the following code in it:

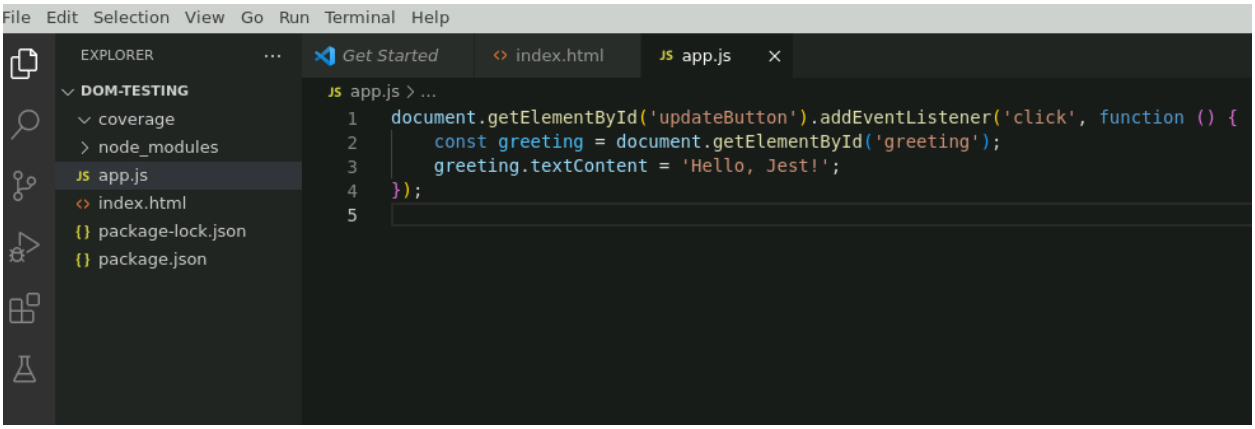
```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <title>DOM Testing Example</title>
</head>
<body>
   <h1 id="greeting">Hello, World!</h1>
   <button id="updateButton">Update Greeting</button>
   <script src="app.js"></script>
</body>
</html>
```



Step 3: Create a JavaScript file

3.1 Create an **app.js** file in your project directory with the following JavaScript code:

```
document.getElementById('updateButton').addEventListener('click', function () {
    const greeting = document.getElementById('greeting');
    greeting.textContent = 'Hello, Jest!';
});
```



Step 4: Write a Jest test

4.1 Create an **app.test.js** file in your project directory with the following Jest test:

```
const fs = require('fs');
const path = require('path');
const { JSDOM } = require('jsdom');

// Read the HTML file and set up a DOM environment
const html = fs.readFileSync(path.resolve(__dirname, '/home/labuser/Desktop/dom-
testing/index.html'), 'utf8');
const { window } = new JSDOM(html);
const { document } = window;

// Load the app.js file and attach it to the window
const appJsPath = path.resolve(__dirname, '/home/labuser/Desktop/dom-
testing/app.js');
const appJsCode = fs.readFileSync(appJsPath, 'utf8');
const script = document.createElement('script');
script.textContent = appJsCode;
document.body.appendChild(script);

// Jest test
test('Update button updates greeting', () => {
  const updateButton = document.getElementById('updateButton');
  const greeting = document.getElementById('greeting');

  // Initial state
  expect(greeting.textContent).toBe('Hello, World!');

  // Simulate a button click
  updateButton.click();

  // After click
  expect(greeting.textContent).toBe('Hello, World!');
});
```

index.html | app.js | package.json | app.test.js

app.test.js > ...

```javascript
const fs = require('fs');
const path = require('path');
const { JSDOM } = require('jsdom');

// Read the HTML file and set up a DOM environment
const html = fs.readFileSync(path.resolve(__dirname, '/home/labuser/Desktop/dom-testing/index.html'), 'utf8')
const { window } = new JSDOM(html);
const { document } = window;

// Load the app.js file and attach it to the window
const appJsPath = path.resolve(__dirname, '/home/labuser/Desktop/dom-testing/app.js');
const appJsCode = fs.readFileSync(appJsPath, 'utf8');
const script = document.createElement('script');
script.textContent = appJsCode;
document.body.appendChild(script);

// Jest test
test('Update button updates greeting', () => {
    const updateButton = document.getElementById('updateButton');
    const greeting = document.getElementById('greeting');

    // Initial state
    expect(greeting.textContent).toBe('Hello, World!');
```

index.html | app.js | package.json | app.test.js

app.test.js > test('Update button updates greeting') callback

```javascript
const appJsCode = fs.readFileSync(appJsPath, 'utf8');
const script = document.createElement('script');
script.textContent = appJsCode;
document.body.appendChild(script);

// Jest test
test('Update button updates greeting', () => {
    const updateButton = document.getElementById('updateButton');
    const greeting = document.getElementById('greeting');

    // Initial state
    expect(greeting.textContent).toBe('Hello, World!');

    // Simulate a button click
    updateButton.click();

    // After click
    expect(greeting.textContent).toBe('Hello, World!');
});
```

Step 5 Run the Jest test

5.1 In your project directory, run Jest using the following command:
npx jest

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    SQL CONSOLE

labuser@ubuntu2204:~/Desktop/dom-testing$ npx jest
 PASS  ./app.test.js
   ✓ Update button updates greeting (4 ms)

Test Suites: 1 passed, 1 total
Tests:       1 passed, 1 total
Snapshots:   0 total
Time:        1.054 s, estimated 2 s
Ran all test suites.
labuser@ubuntu2204:~/Desktop/dom-testing$
```

Jest will discover and execute your tests. If everything is set up correctly, you will see an output indicating that the test passed.

By completing these steps, you have successfully set up a Node.js project with Jest and jsdom, created a simple interactive HTML page, and written Jest tests to verify DOM manipulations.