

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

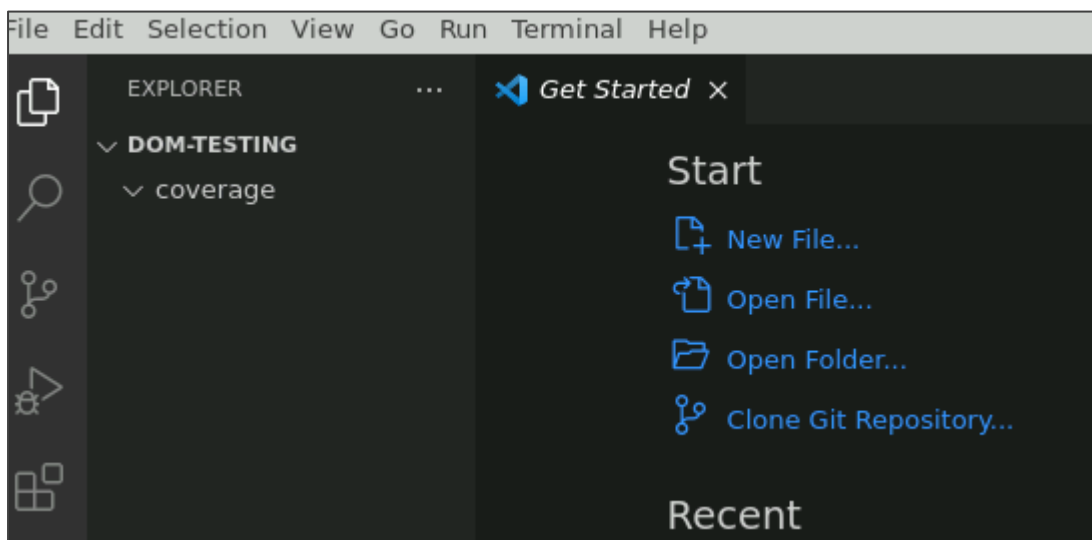
Prerequisites: Good understanding of JavaScript, HTML, DOM, and some knowledge of Jest

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

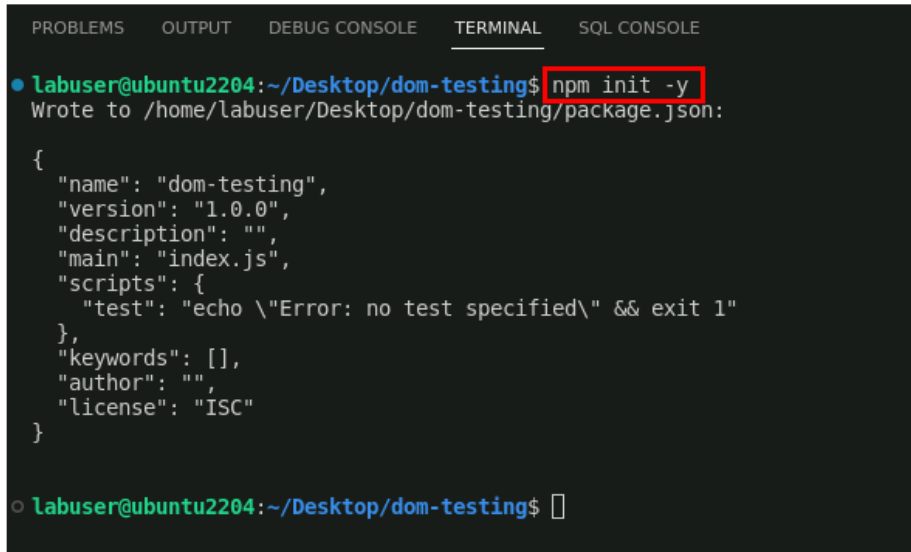
Step 1: Create a Node project

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



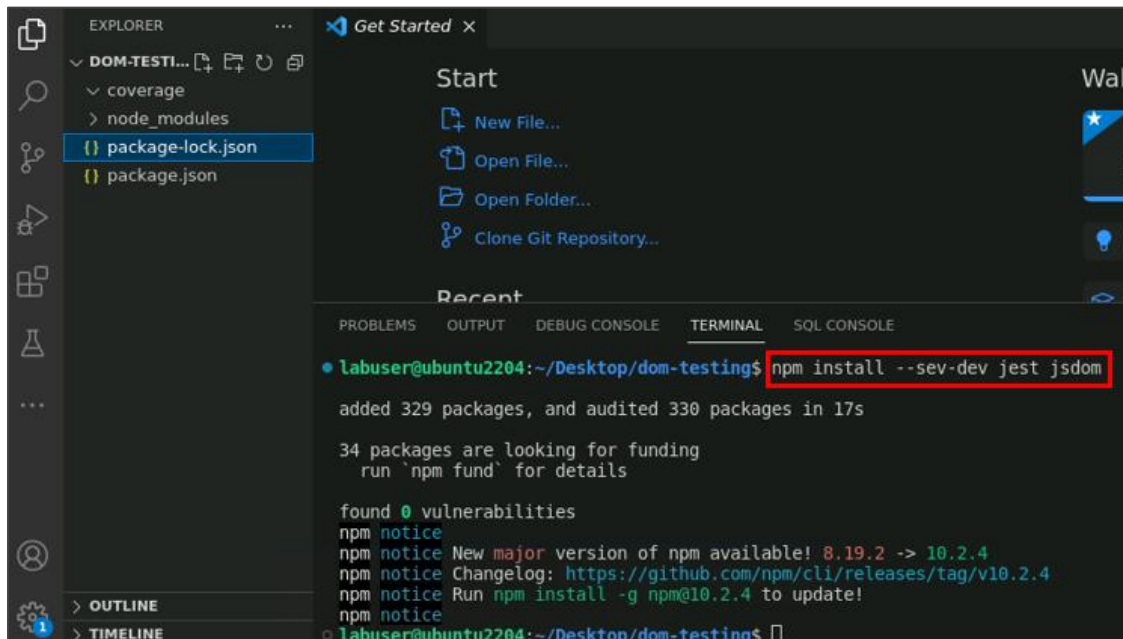
```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL SQL CONSOLE
• labuser@ubuntu2204:~/Desktop/dom-testing$ npm init -y
Wrote to /home/labuser/Desktop/dom-testing/package.json:

{
  "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



```
EXPLORER
  DOM-TESTI...
  coverage
  node_modules
  {} package-lock.json
  {} package.json

Get Started x
Start
  New File...
  Open File...
  Open Folder...
  Clone Git Repository...

Recent
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL SQL CONSOLE
• labuser@ubuntu2204:~/Desktop/dom-testing$ npm install --save-dev jest jsdom
added 329 packages, and audited 330 packages in 17s

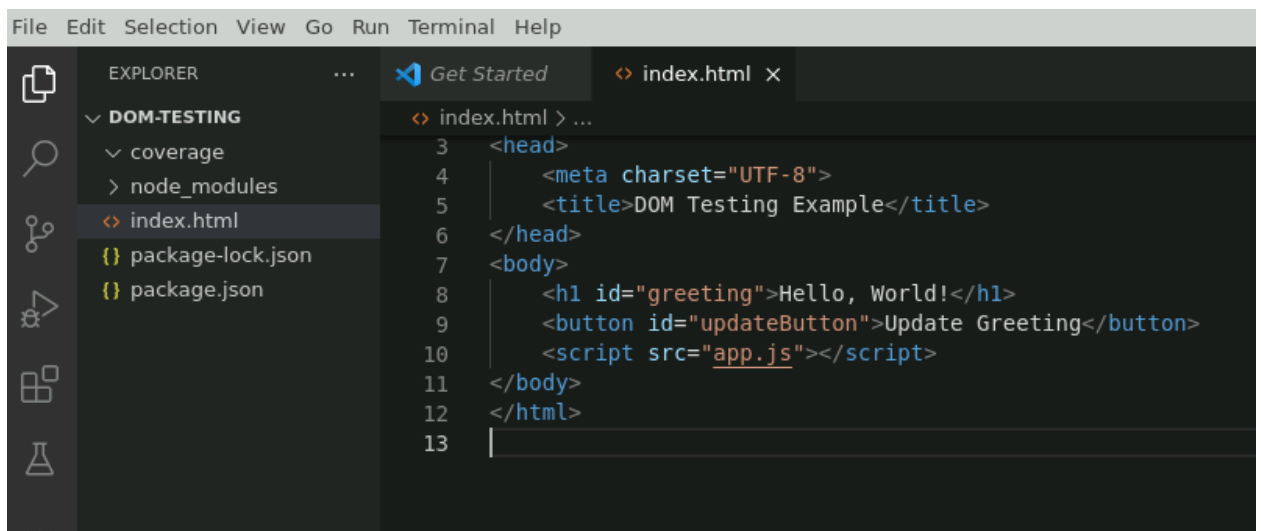
34 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
npm notice
npm notice New major version of npm available! 8.19.2 -> 10.2.4
npm notice Changelog: https://github.com/npm/cli/releases/tag/v10.2.4
npm notice Run npm install -g npm@10.2.4 to update!
npm notice
○ labuser@ubuntu2204:~/Desktop/dom-testing$
```

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:

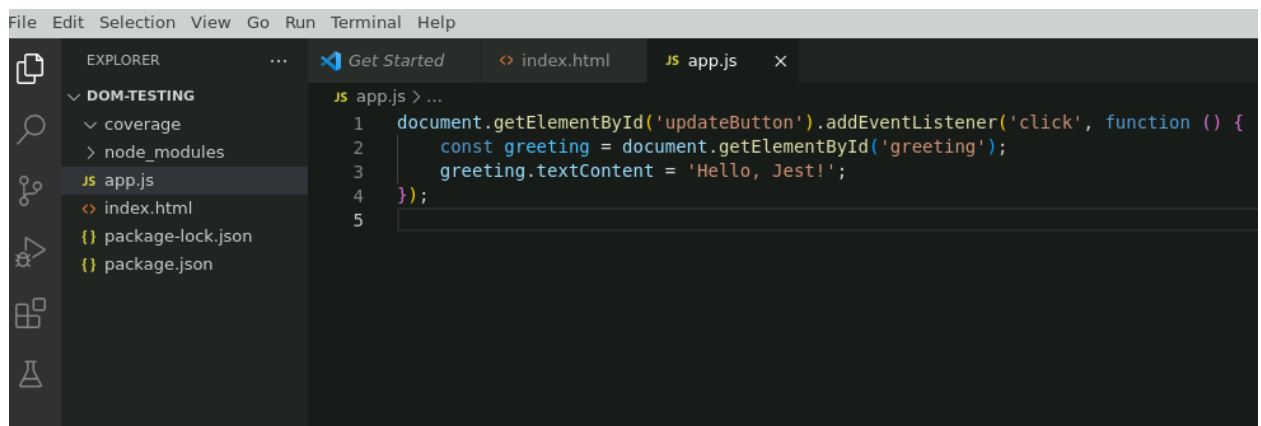
```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>DOM Testing Example</title>
</head>
<body>
  <h1 id="greeting">Hello, World!</h1>
  <button id="updateButton">Update Greeting</button>
  <script src="app.js"></script>
</body>
</html>
```



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  JS app.js  {} package.json  JS app.test.js x
JS app.test.js > ...
1  const fs = require('fs');
2  const path = require('path');
3  const { JSDOM } = require('jsdom');
4
5  // Read the HTML file and set up a DOM environment
6  const html = fs.readFileSync(path.resolve(__dirname, '/home/labuser/Desktop/dom-testing/index.html'), 'utf8')
7  const { window } = new JSDOM(html);
8  const { document } = window;
9
10 // Load the app.js file and attach it to the window
11 const appJsPath = path.resolve(__dirname, '/home/labuser/Desktop/dom-testing/app.js');
12 const appJsCode = fs.readFileSync(appJsPath, 'utf8');
13 const script = document.createElement('script');
14 script.textContent = appJsCode;
15 document.body.appendChild(script);
16
17 // Jest test
18 test('Update button updates greeting', () => {
19   const updateButton = document.getElementById('updateButton');
20   const greeting = document.getElementById('greeting');
21
22   // Initial state
23   expect(greeting.textContent).toBe('Hello, World!');

```

```

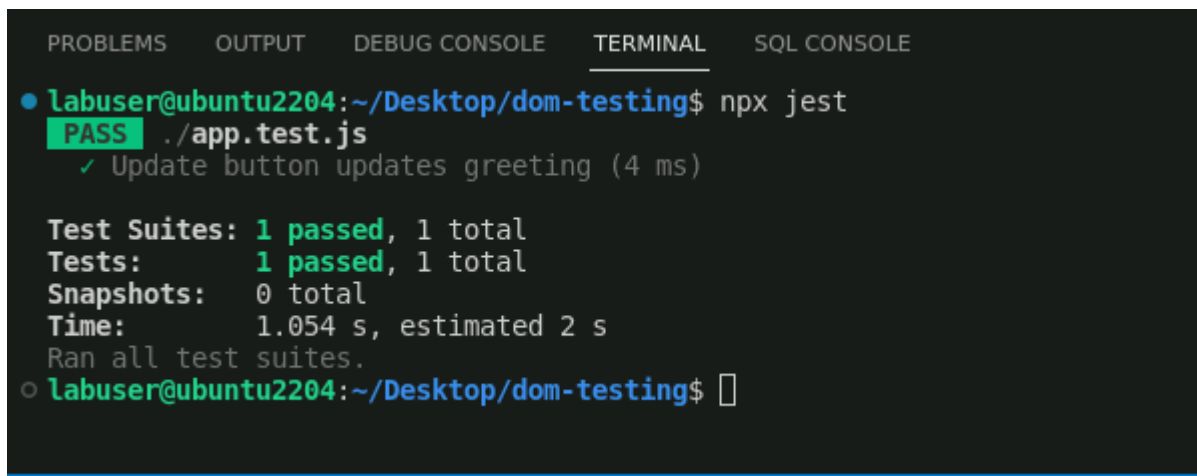
index.html  JS app.js  {} package.json  JS app.test.js ●
JS app.test.js > test('Update button updates greeting') callback
11 const appJsPath = path.resolve(__dirname, '/home/labuser/Desktop/dom-testing/app.js');
12 const appJsCode = fs.readFileSync(appJsPath, 'utf8');
13 const script = document.createElement('script');
14 script.textContent = appJsCode;
15 document.body.appendChild(script);
16
17 // Jest test
18 test('Update button updates greeting', () => {
19   const updateButton = document.getElementById('updateButton');
20   const greeting = document.getElementById('greeting');
21
22   // Initial state
23   expect(greeting.textContent).toBe('Hello, World!');
24
25   // Simulate a button click
26   updateButton.click();
27
28   // After click
29   expect(greeting.textContent).toBe('Hello, World!');
30 });
31
32

```

Step 5 Run the Jest test

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

npx jest

A terminal window with a dark background and light-colored text. At the top, there are tabs for 'PROBLEMS', 'OUTPUT', 'DEBUG CONSOLE', 'TERMINAL' (which is active), and 'SQL CONSOLE'. The terminal shows a command prompt for 'labuser@ubuntu2204' in the directory '~/Desktop/dom-testing'. The command 'npx jest' has been executed. The output shows 'PASS ./app.test.js' with a green checkmark, followed by a list of test results: 'Update button updates greeting (4 ms)'. Below this, a summary is shown: 'Test Suites: 1 passed, 1 total', 'Tests: 1 passed, 1 total', 'Snapshots: 0 total', and 'Time: 1.054 s, estimated 2 s'. It also says 'Ran all test suites.' The prompt returns to the shell.

```
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.