Frontend - Angular and Unit Testing in Angular – Guided exercise

Table of Contents

1.1 Description 2
1.2 Problem statement 2
1.3 Software required 2
1.4 Implementation Steps 3
Step 1: Setting up the Testing Environment 3
Step 2: Writing a Cypress Test 3
Step 3: Setting up Karma and Jasmine 4
Step 4: Writing a Karma/Jasmine Test 5
1.5 Supporting References 6

1.1 Description

Unit testing in Angular refers to the process of testing individual units of code. An Angular unit test aims to uncover issues such as incorrect logic, misbehaving functions, etc. by isolating pieces of code. Performing comprehensive Angular unit testing is critical for several reasons: like code quality assurance, instant feedback loop, improved scalability, reduced maintenance overhead. Angular is designed to help you write code in such a way that enables you to test your app's functions individually in isolation.

To accomplish this task effectively, you can use Angular unit testing frameworks like Karma, Jasmine, and Cypress (formerly known as Codename) along with relevant plugins required for running tests in multiple browsers and debugging environments.

1. **Karma** is a JavaScript test runner that runs the unit test snippet in Angular. Karma also ensures the result of the test is printed out either in the console or in the file log. Karma checks the current state of code under test against any asserted expectations, providing immediate reports once those conditions aren't met. This helps to identify areas needing attention quickly without having to rely solely on human observation alone saving valuable development hours.
2. **Jasmine** is a Behavior Driven Development (BDD)-style syntax for writing specifications or tests with a focus on defining how the program should behave, making them easier for both humans and machines to understand. Its flexibility offers better customization options for specific requirements or libraries, which ultimately leads to clearer, concise, organized, and readable code resulting from improved testing practices.
3. **Cypress** is a modern UI-oriented frontend testing tool built specifically for Angular applications using TypeScript. Since Cypress interacts directly with real DOM elements and dynamic data returned by API calls like actions taken via Selenium WebDriver, it works seamlessly throughout entire apps from the UI layer down to lower-level components. This feature permits developers to write fast, reliable, and isolated tests in conjunction with easy management of their projects thanks to built-in commands simplifying workflows between local dev environments and CI pipelines.

This is guided exercise provided for My Learning Studio(MLS) Learning path - Frontend - Angular and Unit Testing in Angular

By the end of this Learning pathway, the associates will learn the below concepts,

- Unit testing with Karma, Jasmine and Cypress

This Guided exercise helps learners to practice hands-on exercise aligned to topics covered in the above learning path.

1.2 Problem statement

- Test an Angular *hello-world* app using Karma, Jasmine, and Cypresss

1.3 Software required

- Visual Studio Code
- Node.js
- NPM Package Manager

1.4 Implementation Steps

Step 1: Setting up the Testing Environment

1. Create a new Angular project using the Angular CLI:

```
ng new hello-world-appng new hello-world-app
```

2. Navigate to the project directory:

```
cd hello-world-app
```

3. Install Cypress as a dev dependency:

```
npm install cypress --save-dev
```

4. Update the **scripts** section in the **package.json** file to include the Cypress commands:

```
"scripts": {
  "start": "ng serve",
  "cypress:open": "cypress open"
}
```

5. Open the Cypress test runner by running the following command:

```
npm run cypress:open
```

Step 2: Writing a Cypress Test

1. In the Cypress test runner, create a new test file named **hello-world.spec.ts** under the **cypress/integration** directory.

2. Write a simple test to verify that the hello world message is displayed on the page:

```
describe('HelloWorldComponent', () => {
  beforeEach(() => {
    cy.visit('/');
  });
  it('should display hello world message', () => {
    cy.get('h1').should('contain', 'Hello, World!');
  });
});
```

3. Save the file, and the Cypress test runner should automatically detect and run the test.

Step 3: Setting up Karma and Jasmine

1. Install Karma and Jasmine as dev dependencies:

```
npm install karma jasmine karma-jasmine karma-chrome-launcher --save-dev
```

2. Generate a Karma configuration file:

```
npx karma init
```

3. Answer the prompts to configure Karma:

- Select "Jasmine" as the testing framework.
- Choose "Chrome" as the browser.
- Specify the location of your test files (e.g., **src/**/*.spec.ts**).
- Save the generated configuration file (**karma.conf.js**).

4. Update the **scripts** section in the **package.json** file to include the Karma command:

```
"scripts": {
    "start": "ng serve",
    "cypress:open": "cypress open",
    "test": "karma start karma.conf.js"
}
```

Step 4: Writing a Karma/Jasmine Test

1. Create a new test file named **hello-world.spec.ts** under the **src** directory.

2. Write a test to verify that the hello world message is displayed on the page:

```
import { TestBed, ComponentFixture } from '@angular/core/testing';
import { HelloWorldComponent } from './hello-world.component';

describe('HelloWorldComponent', () => {
 let component: HelloWorldComponent;
 let fixture: ComponentFixture<HelloWorldComponent>;
 beforeEach(async () => {
  await TestBed.configureTestingModule({
   declarations: [HelloWorldComponent],
  }).compileComponents();
 });
 beforeEach(() => {
  fixture = TestBed.createComponent(HelloWorldComponent);
  component = fixture.componentInstance;
  fixture.detectChanges();
 });

 it('should display hello world message', () => {
  const compiled = fixture.nativeElement;
  expect(compiled.querySelector('h1').textContent).toContain('Hello,
  World!');
 });
```

3. Run the Karma tests using the following command:

```
npm test
```

1.5 Supporting References

Refer the below links for additional information:

1. Angular Testing documentation
2. Angular Component Testing with Cypress