**Introduction**

In Angular 16, Karma has been deprecated while Jest has been introduced in experimental mode. This means that the Angular team is moving towards Jest as the recommended testing framework for Angular applications. While Karma has been the default testing framework for Angular in the past, Jest is being explored as a potential replacement due to its simplicity, speed, and built-in features like snapshot testing and mocking. However, it’s important to note that Jest is still in the experimental stage in Angular 16, and using it in production may have limitations or unsupported features. Developers should stay informed about the latest updates and recommendations from the Angular team regarding testing frameworks in Angular 16.In this article, we will explore how to add Jest as the testing framework in Angular 16

**Jest:**

Jest is a popular testing framework from Facebook, widely used in the React ecosystem. It provides everything needed for unit testing JavaScript applications, including powerful mocking, built-in assertions, code coverage, and headless testing with JSDOM. With a better developer experience and faster execution compared to Karma and Jasmine, Jest is a comprehensive toolkit for unit testing in JavaScript.

Jest, a testing framework from Facebook, offers several powerful features:

1. Powerful mocking capabilities: Jest provides built-in mocking functionalities that allow you to easily create mock implementations of dependencies. This enables you to simulate different scenarios and isolate units of code for effective testing.
2. Built-in assertion library: Jest comes with its own assertion library, eliminating the need for additional assertion libraries. It provides a wide range of matchers and assertion methods to ensure your code behaves as expected.
3. Built-in code coverage generator: Jest has a built-in code coverage tool that generates reports to show the percentage of code covered by your tests. This helps identify areas of your codebase that may require additional testing.
4. JSDOM for headless testing: Jest leverages JSDOM, a library that simulates the browser’s DOM environment. This enables you to run your tests in a headless manner, without the need for a browser or extra configuration. It simplifies running tests on CI servers and ensures consistent test execution.

I prefer Jest over Karma and Jasmine because I think Jest provides a better developer experience.

1. Jest is faster than Karma, offering optimized test execution that reduces testing time and provides quicker feedback during development.
2. Jest is well-documented, providing extensive resources and examples that make it easy for developers to get started and effectively use the framework.
3. Jest is smarter than Karma as it reruns only affected tests, detecting changes in code and selectively executing relevant tests, improving overall test execution efficiency.
4. Jest is a complete toolkit for unit testing, providing powerful mocking capabilities, built-in assertions, code coverage generation, and additional features that streamline the testing process.
5. Jest is headless by default with JSDOM, allowing tests to run without the need for a browser, making it suitable for headless testing scenarios and facilitating integration with CI/CD pipelines.

**Jest setup in an Angular project**

**Step 1: Create new project by using this command:**

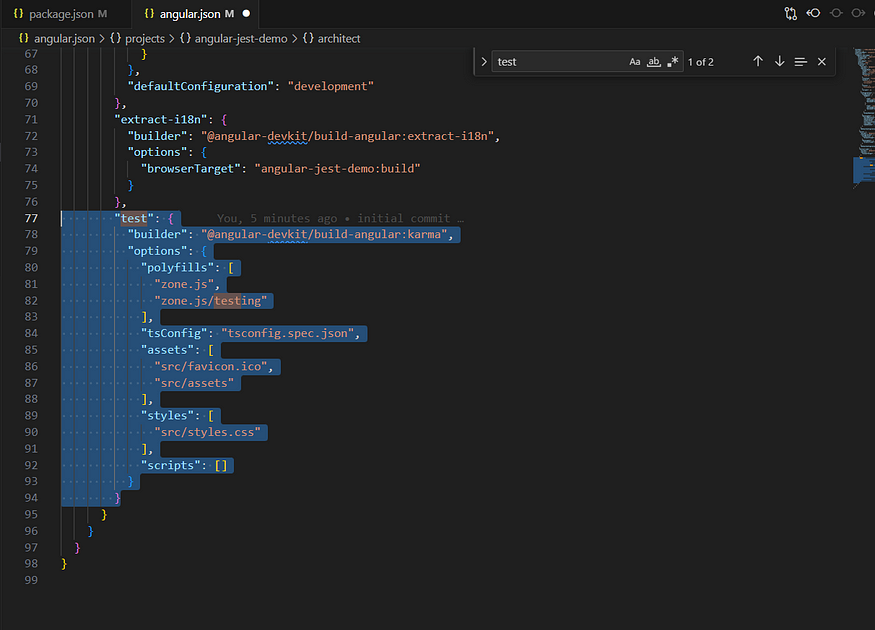
ng new angular-jest-demo

**Step 2: Uninstall all karma jasmin packages by using below command:**

npm uninstall karma karma-chrome-launcher karma-coverage karma-jasmine karma-jasmine-html-reporter @types/jasmine jasmine-core



**Step 3: Remove test object from angular.json file**



**Step 4: Install below mentioned packages:**

npm i --save-dev jest @types/jest jest-preset-angular

The @types/jest package provides type declaration files for Jest, enabling TypeScript to perform type checking and provide IntelliSense support for Jest-specific APIs, functions, and matchers.

The jest-preset-angular package is a crucial tool for running Angular unit tests with Jest. It provides the following features:

1. ts-jest: A library included in @types/jest that allows Jest to transpile TypeScript code in-memory, facilitating seamless integration between TypeScript and Jest during testing.
2. Snapshot serializers: @types/jest provides specialized serializers for Angular components, enhancing snapshot testing capabilities by enabling easy comparison and validation of component snapshots.
3. AST transformers: The included AST transformers in @types/jest modify Angular component code during testing, removing CSS styles and inlining HTML templates. This ensures compatibility with JSDOM, enabling headless testing without relying on external resources.

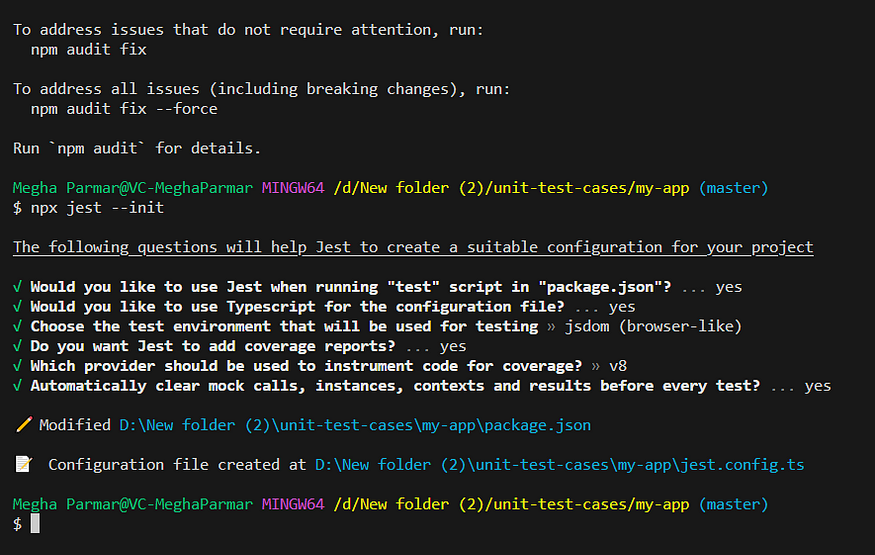
**step 5 : Create setup-jest.ts file in root folder**

import 'jest-preset-angular/setup-jest';

**step 6 : To create jest.config file use below command:**

npx jest — init

The npx jest - init command initializes Jest in your project by creating a jest.config.js / jest.config.ts file and providing a guided setup process



Set preset and setupFilesAfterEnv value in jest.config.ts file which are mentioned on below:

preset: 'jest-preset-angular',  
setupFilesAfterEnv: ['<rootDir>/setup-jest.ts'],

We are require to install ts-node for the TypeScript configuration files

*npm i ts-node*

ts-node is provides a convenient way to execute TypeScript files and experiment with TypeScript code interactively.

**Step 7 : Update tsconfig.spec.json file**

/\* To learn more about this file see: https://angular.io/config/tsconfig. \*/  
{  
 "extends": "./tsconfig.json",  
 "compilerOptions": {  
 "outDir": "./out-tsc/spec",  
 "types": [  
 "jest" // 1  
 ],  
 "esModuleInterop": true, // 2  
 "emitDecoratorMetadata": true // 3  
 },  
 "include": [  
 "src/\*\*/\*.spec.ts",  
 "src/\*\*/\*.d.ts"  
 ]  
}

1. Register Jest’s type definitions files with the TypeScript compiler.
2. Enable the esModuleInterop option of the TypeScript compiler otherwise Jest will output a lot of warnings in the console.
3. Enable the emitDecoratorMetadata option of the TypeScript compiler otherwise Angular's Dependency Injection won't work with Jest.

**step 8: Add jest configuration on package.json inside script tag**

"test": "jest --verbose",  
 "test:coverage": "jest --coverage",  
 "test:watch": "jest --watch"

**Step 9: Write a unit test cases**

import { ComponentFixture, TestBed } from '@angular/core/testing';  
import { AppComponent } from './app.component';  
  
describe('AppComponent', () => {  
 let component: AppComponent;  
 let fixture: ComponentFixture<AppComponent>;  
  
 beforeEach(async () => {  
 await TestBed.configureTestingModule({  
 declarations: [AppComponent],  
 }).compileComponents();  
 });  
  
 beforeEach(() => {  
 fixture = TestBed.createComponent(AppComponent);  
 component = fixture.componentInstance;  
 fixture.detectChanges();  
 });  
  
 it('should create the app', () => {  
 expect(component).toBeTruthy();  
 });  
  
 it(`should have as title 'demo-angular-jest'`, () => {  
 expect(component.title).toEqual('demo-angular-jest');  
 });  
  
 it('should render the title', () => {  
 const compiled = fixture.nativeElement as HTMLElement;  
 expect(compiled.querySelector('.content span')?.textContent).toContain('demo-angular-jest app is running!');  
 });  
});

