Angular unit testing with Jest

Harry Kalligeros

Senior Frontend Developer @ Trasys Greece

What is a unit ?

A unit is the smallest testable piece of code, such as a function, class, procedure, interface.

What is a unit testing ?

- A software testing method by which individual units are tested to determine whether they are fit for use
- It assures that when a set of inputs are given, the unit test should return the expected values

Why tests are useful ?

- They are a written contract, that the code must satisfy
- They protect against changes that break existing code (regressions)
- They help building a solid software design
- They provide a living documentation of the system

F.I.R.S.T. Principles of Unit Testing

- Fast
- Isolated
- Repeatable
- Self-verifying
- Timely

Fast

- A developer should not hesitate to run
- All of these including setup, the actual test and tear down should execute really fast (milliseconds) as you may have thousands of tests in your entire project.

Isolated/Independent

- A test method should do the **3 As => Arrange, Act, Assert**
- **Arrange**: The data used in a test should not depend on the environment in which the test is running.
- **Act**: Invoke the actual method under test.
- **Assert**: A test method should test for a single logical outcome, implying that typically there should be only a single logical assert.
- No order-of-run dependency. They should pass or fail the same way in suite or when run individually

Repeatable

- A test method should NOT depend on any data in the environment/instance in which it is running.
- Deterministic results - should yield the same results every time and at every location where they run. No dependency on date/time or random functions output.
- Each test should setup or arrange it's own data. What if a set of tests need some common data? Use Data Helper classes that can setup this data for re-usability.

Self-validating

- No manual inspection required to check whether the test has passed or failed.

Thorough/Timely

- Cover every use case scenario and NOT just aim for 100% coverage
- Tests for corner/edge/boundary values.

Angular testing utilities

TestBed

- Most important testing utility
- Creates a dynamically-constructed Angular test module that emulates an Angular *@NgModule*.
- `TestBed.configureTestingModule()` method takes a metadata object that can have most of the properties of an *@NgModule*.
- More info... https://angular.io/guide/testing

Integrating Jest into Angular CLI

- Follow the instructions: https://blog.angularindepth.com/integrate-jest-into-an-angular-application-and-library-163b01d977ce

Jest

- Jest is a delightful JavaScript Testing Framework with a focus on simplicity.
- Fast and safe
- Easy mocking
- Code coverage

https://jestjs.io

Mocking

Mocking is a technique to isolate test subjects by replacing dependencies with objects that you can control and inspect

Jest Mock Function

The Mock Function provides features to:

- Capture calls
- Set return values
- Change the implementation

The simplest way to create a Mock Function instance is with `jest.fn()`.

Spies

- Jest Spies watch a method invocation of an object and leave the original implementation in place
- Usage:

```
const spy = jest.spyOn(object, 'methodName');
```

- Assertions:

```
expect(spy).toHaveBeenCalled();
expect(spy).toHaveBeenCalledWith(arg1, arg2);
```

Resources

- https://github.com/ghsukumar/SFDC_Best_Practices/wiki/P
  Principles-of-Unit-Testing
- https://angular.io/guide/testing
- https://jestjs.io/docs/en/getting-started
- https://blog.angularindepth.com/integrate-jest-into-an-angul
  application-and-library-163b01d977ce
- https://itnext.io/how-to-use-jest-in-angular-aka-make-unit-te
  great-again-e4be2d2e92d1