Angular - Unit Testing
Jasmin to Jest

unit testing
mocking
Jasmine + karma
Experiment + Problem
Solution = jest + spectator
Experiment
*glance over Integration Testing - cypress

Unit Test:

It’s where individual unit(s) or function(s) or component(s) is tested.
It’s you individually functioning within your team.
Still you have look at the broader scope.

By Book:

It follows SRP.

https://martinfowler.com/bliki/UnitTest.html
https://en.wikipedia.org/wiki/Unit_testing

experiment-1

simple unit test

```
add(n1: number, n2: number) {
  this.logger.log('Addition operation called');
  return n1 + n2;
}

subtract(n1: number, n2: number) {
  this.logger.log('Subtraction operation called');
  return n1 - n2;
}
```

Stub | Mock:

Mission Impossible - Interceptors/Agents.
It handles the calls itself without invoking them.
Imitation of an original Object.

good Read:

https://martinfowler.com/articles/mocksArentStubs.html

Jasmine
&
Karma

Jasmine-
JavaScript Testing Framework.
Easy to write Syntax.

https://jasmine.github.io/
https://codecraft.tv/courses/angular/unit-testing/jasmine-and-karma/
https://dev.to/mustapha/angular-unit-testing-101-with-examples-6mc
https://medium.com/@pjbgf/title-testing-code-ocd-and-the-aaa-pattern-df453975ab80

Jasmine -

Provides structural support :)
Arrange, Act, Assert (AAA)

- describe () - test suite
- it() - test specification
- expect() - result, Assert

beforeEach, afterEach, spyOn

Karma-

This is where unit tests are running. It's a test runner.

https://www.softwaretestinghelp.com/karma-test-runner-tutorial/

https://codecraft.tv/courses/angular/unit-testing/jasmine-and-karma/
https://karma-runner.github.io/latest/index.html

The Experiment

Simple Test
Component Test
Test with Mock

The Problem
with Jasmine & Karma

It launches THE BROWSER and
slows down testing.
this the complaint of Jest
Supporters.
for HG - it was a lot of setup.

Jest - js testing framework

spectator - remove boilerplate code

https://jestjs.io/en/
https://github.com/ngneat/spectator

Jest - Why another framework?

Jest offers some benefits like...

- it’s fast
- easy mocking*
- code coverage
- Most valuable >> It does not need THE BROWSER instance to run tests.

Jest

- describe () - test suite
- test() - test specification
- expect() - result, Assert

beforeEach, afterEach,

let mockObject = jest.fn(function) << simple

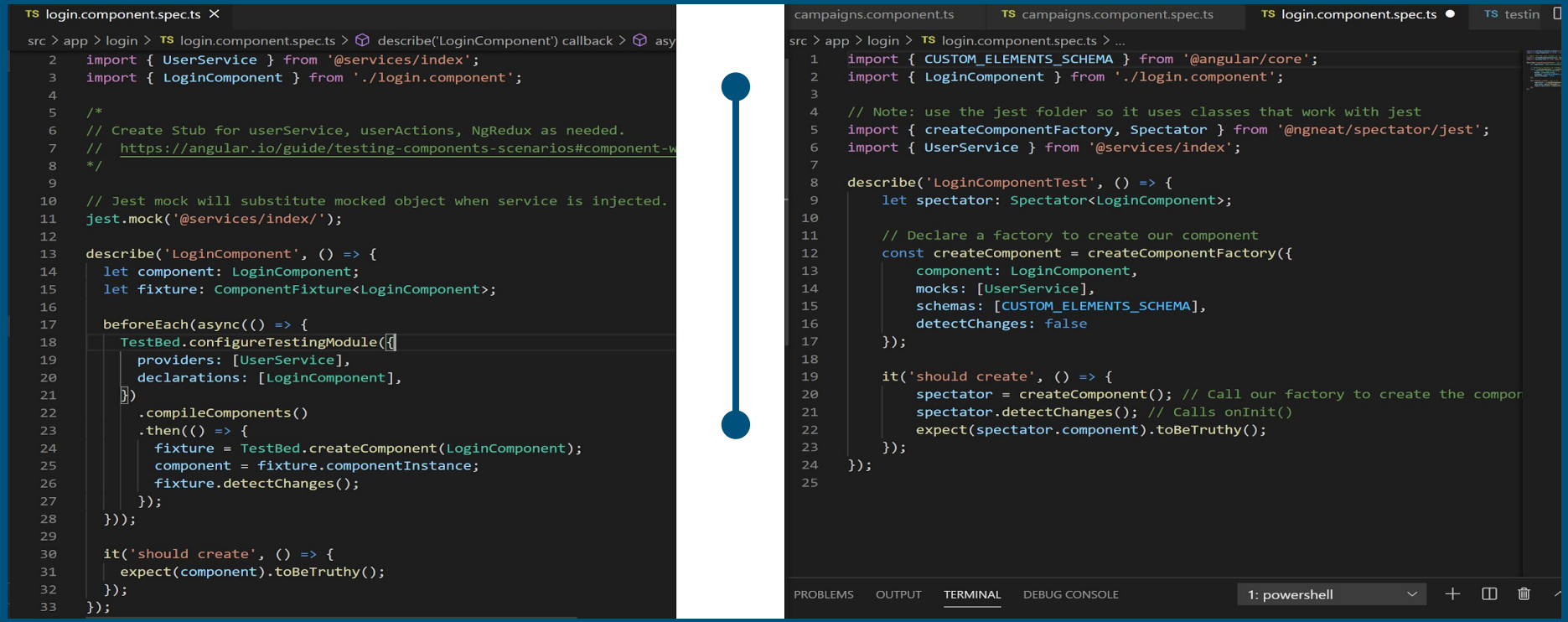
let mockModule = jest.mock(module)

https://medium.com/@rickhanlonii/understanding-jest-mocks-f0046c68e53c

npx jest

npx jest --login.component.spec.ts

Spectator makes difference.



login.component.spec.ts

```
src > app > login > login.component.spec.ts > describe('LoginComponent') callback > asy
import { UserService } from '@services/index';
import { LoginComponent } from './login.component';

/*
// Create Stub for userService, userActions, NgRedux as needed.
//  https://angular.io/guide/testing-components-scenarios#component-w
*/

// Jest mock will substitute mocked object when service is injected.
jest.mock('@services/index/');

describe('LoginComponent', () => {
  let component: LoginComponent;
  let fixture: ComponentFixture<LoginComponent>;

  beforeEach(async(() => {
    TestBed.configureTestingModule({
      providers: [UserService],
      declarations: [LoginComponent],
    })
      .compileComponents()
      .then(() => {
        fixture = TestBed.createComponent(LoginComponent);
        component = fixture.componentInstance;
        fixture.detectChanges();
      });
  }));

  it('should create', () => {
    expect(component).toBeTruthy();
  });
});
```

login.component.spec.ts

```
src > app > login > login.component.spec.ts > ...
import { CUSTOM_ELEMENTS_SCHEMA } from '@angular/core';
import { LoginComponent } from './login.component';

// Note: use the jest folder so it uses classes that work with jest
import { createComponentFactory, Spectator } from '@ngneat/spectator/jest';
import { UserService } from '@services/index';

describe('LoginComponentTest', () => {
    let spectator: Spectator<LoginComponent>;

    // Declare a factory to create our component
    const createComponent = createComponentFactory({
        component: LoginComponent,
        mocks: [UserService],
        schemas: [CUSTOM_ELEMENTS_SCHEMA],
        detectChanges: false
    });

    it('should create', () => {
        spectator = createComponent(); // Call our factory to create the compor
        spectator.detectChanges(); // Calls onInit()
        expect(spectator.component).toBeTruthy();
    });
});
```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE 1: powershell

Closing remarks

Why all the hassle? setup unit test, mock, jasmine to jest, why????

Think about future….. as of now you have control, idea, knowledge of how requirements should function, a hold of code that you are writing.

You will set expectation at best right now and if you or someone changes same code and it’s not compatible. It will fail test.

You have a chance to catch that error before it goes to production. hurray!

Resources

https://angular-university.io/

https://dev.to/mustapha/angular-unit-testing-101-with-examples-6mc

https://codecraft.tv/courses/angular/unit-testing/jasmine-and-karma/

https://jestjs.io/docs/testing-frameworks

https://www.xfive.co/blog/testing-angular-faster-jest/

Just in case

If you need me...

hitesh.ghori@sogeti.com

(412) 726 3821: Text/Call is fine

https://www.linkedin.com/in/hitesh-ghori-735b8515/