## Jasmine :

## >open source JS testing framework.

## >Used to write test cases for any JS application

## >It is BDD (behavior driven development)

## BDD:

## >It is written in non-technical language so everyone can understand it easily)

## >It mainly focus on the testing the behavior rather than implementation

## >supports asynchronous testing

## Karma:

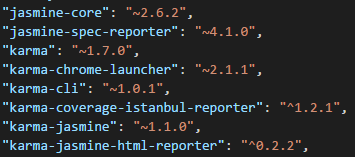
## >it is an open source and test automation tool created by angular JS team

## >karma tool used to run JavaScript test cases. It allow us to execute the test cases on any browser.

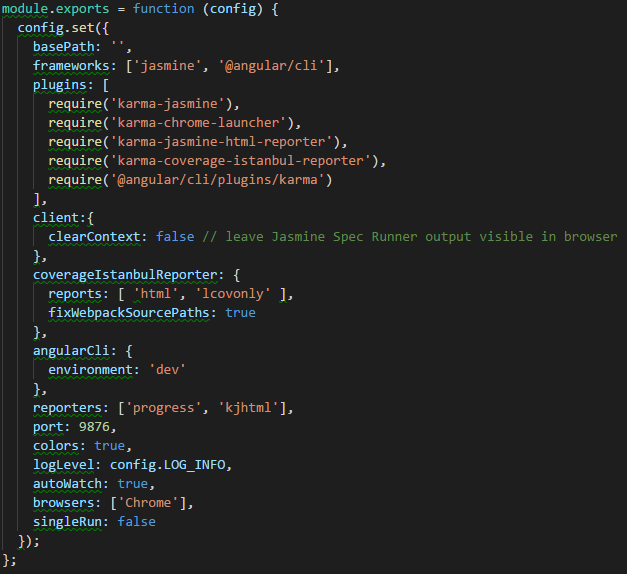
## >any browser can be used and angular cli allows us to run testcases on any browser.

## Create an Angular project with jasmine and karma

As the angular team recommends we are going to use [angular-cli](https://cli.angular.io/) to create our app. By doing this the configuration of jasmine and karma comes resolved for us.



* **jasmine-core.**[Jasmine](https://jasmine.github.io/) is the testing framework which we are going to use to create our tests. It has a bunch of functionalities to allow us the write different kinds of tests.
* **karma.**[Karma](https://karma-runner.github.io/1.0/index.html) is a task runner for our tests. It uses a configuration file in order to set the startup file, the reporters, the testing framework, the browser among other things.
* The rest of the dependencies are mainly reporters for our tests, tools to use karma and jasmine and browser launchers.



* **frameworks**: this is where jasmine gets set as a testing framework. If you want to use another framework this is the place to do it.
* **reporters**: this is where you set the reporters. You can change them or add new ones.
* **autoWatch**: if this is set to true, the tests run in watch mode. If you change any test and save the file the tests are re-build and re-run.
* **browsers:**this is where you set the browser where the test should run. By default it is chrome but you can install and use other browser launchers.

**Test entry file**

* The angular-cli configuration of karma uses the file “test.ts” as the entry point of the tests for the application.



* An environment to run angular tests is being created using all the imports at the beginning of the file.
* TestBed is a powerful unit testing tool provided by angular, and it is initialized in this file.
* Finally, karma loads all the test files of the application matching their names against a regular expression. All files inside our app folder that has “spec.ts” on its name are considered a test.

To run the test you only need to run the command “ng test”. This command is going to execute the tests, open the browser, show a console and a browser report and, not less important, leave the execution of the test in watch mode.

>If you want to run specific folder test case

ng test --include src\app

Exclude/Include Angular Unit test case or test Suite from Execution:

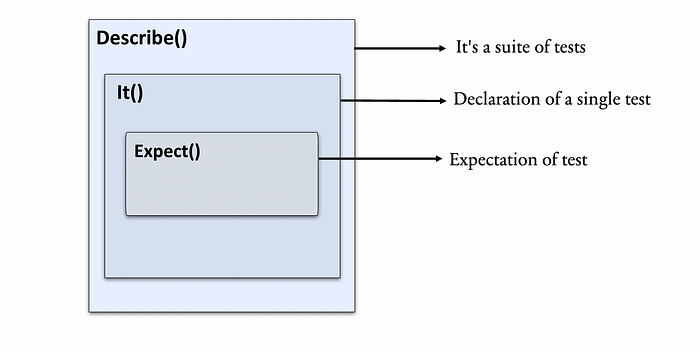
Pending states --- It represents like pending state

**X-**🡪 in front of it/describe 🡪exclude

f->same ---Include

How to write a unit test with Jasmine?

To write a unit test, we need to create a **spec.ts**file or test case file. Generally, Angular generates a test case file on its own and provides a basic skeleton of the test case.



//to group all related test cases

Describe()---2 arguments---“name of the suite”, autonomous function

//It is used to define a single test spec

It()-2 arguments---“name of the test”, autonomous function

1. **Expect()**: It is used to create an expectation or assertion in a test spec. It tests the value against the expected value.

Expect(Which you want to test like properties,methods etc).matcher function(value)

Matchers in angular unit test case:

Check the output by passing input value

>matchers are functions used to compare the expected and actual result in the test spec.

Expect(actual result).matcher function(expected value)

2types:

>in-built matchers( comes with jasmine framework)

>custom matchers (our own matcher defining)

>toBe -----primitives like strings,numbers,Booleans

>toEqual ----arrays, objects

>toBe(true)

Actual === expected

>toBeTrue()

>toBeFalse()

>toBeTruthy()

>toBeGreaterThan

>toBegreaterThanOrEqual

>toBeLessThan

>toBeLessThanOrEqual

>toMatch()---applied for regular expressions

Expect().toMatch()

Expect().not().toMatch()

>toBeDefined

>toBeUndefined---

>tobenull --- to check whether it is null or not

>tocontain---to find an item in an array, to check for string

The execution of one test should not impact the other, so we don’t want fragile test. (let a = new A() -- if we use same instance it will affect so we need to declare this instantiation in beforeEach, so that every test case(has different instances) runs in an isolated world as if there is nothing in the world)

beforeEach,afterEach --- it runs before each test case and after each test case in a describe() ----reusable properties can be defined here

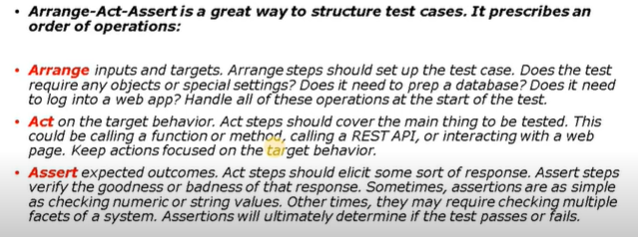
beforeAll,afterAll---It executes once and only once for describe() block containing it, just before any block run

Arrange-Act-Assert pattern---(AAA) patern (setup and tear down)

Arrange: initializations to call method

Act: involves calling a method

Assertion: Testing whether we are getting expected lines or not





**Testbed and Component Fixture:**

Angular test Bed(ATB) is Angular only testing framework which helps us to test behaviors which depends on angular.

Ex: Observables, Input and Output, Services, directives, Pipes, DI, forms, modules, components etc.

We still write our tests in jasmine only and run using karma but we now have an easier way to create components, handle injection. Testing asynchronous behavior and interact with our application.

When to use ATB.

>it allows us to test the interaction of component/directive with the template

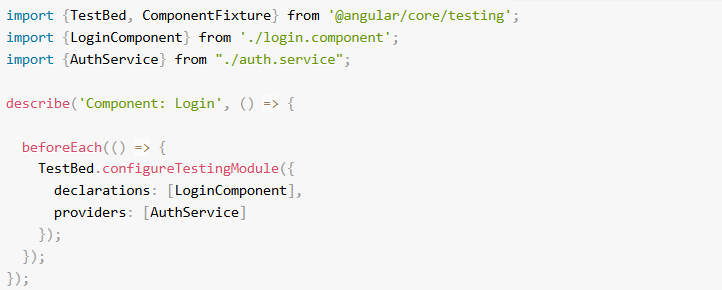
>to test change detection

>to create components, pipes, services, directives, modules etc

>to test and use DI

>to test metadata of module i.e. Ng Module configuration which we use in our application.

>to test user interaction via clicks and input fields



TestBed-🡪Class

configureTestingModule()->

Here we need to give what component depends on Like services, modules (forms, Http, our own project modules etc.)

In the beforeEach function for our test suite we configure a testing module using the TestBed class.

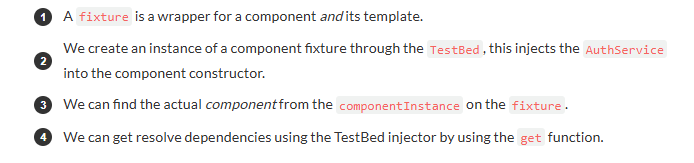
This creates a test Angular Module which we can use to instantiate components, perform dependency injection and so on.

We configure it in exactly the same way as we would configure a normal NgModule. In this case we pass in the LoginComponent in the declarations and the AuthService in the providers.

## [Fixtures and DI](https://codecraft.tv/courses/angular/unit-testing/angular-test-bed/#_fixtures_and_di)

Once the ATB is setup we can then use it to instantiate components and resolve dependencies, like so:





With help of fixture we can access html, css, component.

Fixture helps to see whether class initiated or executed properly or not.

compileComponents()—It compiles all which provided in testBedConfiguration.

# **SpyOn to mock and Stub methods in angular unit test :**

It used to control whether to call origin method or not.

* Calling mock method
* Calling origin method

>to mock the execution of a method

>just from test case point of view, to know whether method got called or not we can check

>we can return our own data with spy (instead of exectuting method)

By using .and.returnValue()

>we can call original method by using .and.callThrough()