Revisit

1. Node
2. NPM
3. Angular CLI
4. Folder Structure of an Angular App
5. What is mean by package.json
6. What is angular.json
7. What is karma.conf.js
8. Script – ng serve, ng test, ng e2e, ng build

Jasmine – Javascript Unit Testing Framework

Different Types of Testing

1. Unit Testing – Testing all the classes, interfaces and each methods of classes
2. Component - .ts, spec.ts, .html, .css (.spec.ts – unit test code)
3. BDD – Behavior Driven Development ( Gherkhin – Simple English words)

Jasmine official Doc - <https://jasmine.github.io/>

describe("A suite is just a function", **function**() {

**var** a;

it("and so is a spec", **function**() {

a **=** **true**;

expect(a).toBe(**true**);

});

});

Junit is for JAVA

Jasmine is for JavaScript

Jasmine stand-alone url -- <https://github.com/jasmine/jasmine/releases>

Test Suite – Group/Collection of test cases – always starts with “describe” with two arguments

//sample Jasmine Test Suite

//First String argument is the “Title of the Test Suite”

//Second argument is a function which describes the test suite.

describe(“Example Test Suite”, function() {

….

});

Test Spec – It represents a test case inside the test suite. Always start with the Global function of Jasmine “it” with two parameters

//First String Parameter “Title of the Spec”

//Second parameter is the function which implements the test case

//Test case will have some expectations that can be checked using “expect”

//Sample test case

It(“My First Jasmine Test Case”, function () {

Expect (expression).toEqual(true);

});

beforeEach() – Called once before each test spec in the describe method – initializing the variables – setup code

afterEach() – Called once after each test spec in the describe method – resetting the variable – tear down

Best practice – Keep all the spec variable defined in top-level-scope

Describe function – is to group all the related specs/test cases.

Describe function can be nested, meaning – you can have describe function inside another describe function

Jasmine Matchers

|  |  |  |
| --- | --- | --- |
| No | Name | Purpose |
| 1 | toBe() | Passed if the actual value is of the same type and value as that of expected value. It compares with === operator |
| 2 | toEqual() | Works for simple literals, variables & objects |
| 3 | toMatch() | To check whether a value matches a string or regular expression |
| 4 | toBeDefined() | To ensure that a property or a value is defined |
| 5 | toBeUndefined() | To ensure that a property or a value is not defined/undefined |
| 6 | toBeNull() | To ensure that a property or a value is “null” |
| 7 | toBeTruthy() | To ensure a property or value is true |
| 8 | toBeFalsy() | To ensure a property or value is false |
| 9 | toContain() | To check whether a string or array contains a substring or an item |
| 10 | toBeLessThan() | For Mathematical comparison of less than |
| 11 | toBeGreaterThan() | For Mathematical comparison of greater than |
| 12 | toBeCloseTo() | For precision math comparison |
| 13 | toThrow() | For testing if a function throws an exception |
| 14 | toThrowError() | For testing a spec thrown expection |

“Not” – keyword can be used with all matchers to invert the result.

expect(actual).not.toBe(expected);

Disable Test suite.

Xdescribe() -- to disable the test suite.

Understanding spies

Spy – Test double function – can stub any function & tracks calls to it and all arguments.

We can add “spy” in describe or it block. It will be removed after each spec.

The method to create a spy is “spyOn(object, ‘methodName’)

Two matchers for Spy

1. toHaveBeenCalled
2. toHaveBeenCalledWith

Creating a Spy with createSpy()

var callback = jasmine.createSpy(‘callback’);

expect(object.callback).toHaveBeenCalled();

tape = jasmine.createSpyObj(‘tape’,[‘play’,’pause’,’stop’,’rewind’]);

tape.play();

expect.(tape.play.calls.any()).toEqual(true);

Setup Jasmine with NPM

1. Local Installation (npm install jasmine) This will install in the current folder. This will be accessible only from the installed folder.
2. Global Installation (npm i -g jasmine) this will install in AppData/Roaming folder by default. This will be accessible from any location

Steps

1: Create a Folder “jasmine ws”

2: Open this folder in VS Code

3: Check Node & NPM version (node -v , npm -v)

4: npm install -g –save-dev jasmine (Add to env variable) -- C:\Users\syska\AppData\Roaming\npm

5: jasmine init (It will add a spec folder in the current location)

6: jasmine (This will run jasmine test)

Testing Angular application with Jasmine

1: Create angular app (ng new angular-testing)

2: Run the angular app (cd angular-testing, ng serve -o)

3: Test the angular app (ng test)

Testing – To Ensure that the developed application/product is working as per the expectation

To prevent software defects

Unit Testing , Integration Testing, Performance Testing, Load Testing, e2e Testing, Black box testing, Grey box testing,

Karma – Is the product of Angular

Karma – is the default test runner for the angular app. (jasmine, Mocha, Qunit, ci tools Jenkins, TravisCi, Circle CI)

The default karma conf – This will run the test in a sperate browser window/ terminal

Jasmine – Is a BDD framework for Unit Testing the JavaScript code

beforeAll

afterAll

1. Testing a Component (done)
2. Testing a Service (json/ api) – Observable (try adding a new service to our angular-test app)
3. Testing a pipe – (only for view purpose) (try adding a new pipe to our angular-test app)