Amit Debadwar

amit.debadwar@gmail.com  9096174175

Complete Angular 7

Contents

[1. Angular Jasmin Karma Unit Testing 2](#_Toc533167325)

[1. Unit Testing 2](#_Toc533167326)

[1. Simple Unit Test 2](#_Toc533167327)

[2. Array Testing 3](#_Toc533167328)

[3. Setup and Tear Down 3](#_Toc533167329)

[4. Reactive Forms Testing 5](#_Toc533167330)

[5. Event Emitter Testing 5](#_Toc533167331)

[6. Creating Service, Components with new keyword and Spying On them 6](#_Toc533167332)

[2. Integration Testing 8](#_Toc533167333)

[1. Testbed 8](#_Toc533167334)

[2. With Async 11](#_Toc533167335)

[**3.** Query **By** 12](#_Toc533167336)

[3. spyOn 14](#_Toc533167337)

[4. Stub 14](#_Toc533167338)

[5. Testing router 14](#_Toc533167339)

[6. Testing router outlet 15](#_Toc533167340)

[7. Attribute Directives Testing 16](#_Toc533167341)

[8. Testing Custom and Structural Directives 17](#_Toc533167342)

[2. Animation (Web API Animations) 18](#_Toc533167343)

[1. fade 18](#_Toc533167344)

[2. Animation with State function 18](#_Toc533167345)

[3. Transition function with Repetition 19](#_Toc533167346)

[4. Transition function repetition avoided with multiple state comma separated. 19](#_Toc533167347)

[5. Bidirectional state in one go 19](#_Toc533167348)

[6. State aliases (:enter, :leave) 20](#_Toc533167349)

[7. Fade in/out animation code 20](#_Toc533167350)

[**8.** Keyframes 21](#_Toc533167351)

[Angular 7 Useful Data 21](#_Toc533167352)

# Angular Jasmin Karma Unit Testing

## Unit Testing

### Simple Unit Test

import { compute } from './compute';

describe('compute', ()=>{

it('should be zero if negative value provided',()=>{

const result = compute(-786);

expect(result).toBe(0);

});

it('should increament the number if input is positive',()=>{

const result = compute(1);

expect(result).toBe(2);

});

})

import { greet } from './greet';

describe('greet',()=>{

it('should include the name in the message', ()=>{

expect(greet('sai')).toContain('sai');

});

});

### Array Testing

import { getCurrencies } from './getCurrencies';

describe('getCurrencies',()=>{

it('should return the supported currencies', ()=>{

//Arrange: Creating system under test

const result= getCurrencies();

//Act

expect(result).toContain('USD');

expect(result).toContain('AUD');

expect(result).toContain('EUR');

});

});

### Setup and Tear Down

import { VoteComponent } from './vote.component';

describe('VoteComponent', () => {

let component : VoteComponent;

/\* In jasmin we have, before each test, beforeEach function will be called,

\* this function is used to

\* initialize the objects

\* As the set up

\*/

beforeEach(() => {

component = new VoteComponent();

});

/\* This function also gets called before each test,

\* here you can clean up the objects

\* tear down

\*/

afterEach(() => {

});

/\* This will be called once before all the tests,

\*

\*/

beforeAll(() => {

});

/\* This will be called once after all the tests,

\*

\*/

afterAll(() => {

});

it('should increament vote when upVoted', () => {

//Arrange : Creating system under test

//Act : Calling a method or function

component.upVote();

//Assert

expect(component.totalVotes).toBe(1);

});

it('should decreament vote when downVoted', () => {

//Arrange : Creating system under test

//Act : Calling a method or function

component.downVote();

//Assert

expect(component.totalVotes).toBe(-1);

});

it('should d', () => {

});

});

### Reactive Forms Testing

import { FormBuilder } from '@angular/forms';

import { TodoFormComponent } from './todo-form.component';

describe('TodoFormComponent', () => {

var component: TodoFormComponent;

beforeEach(() => {

component = new TodoFormComponent(new FormBuilder());

});

it('should create a form with 2 controlls', () => {

expect(component.form.contains('name')).toBeTruthy();

expect(component.form.contains('email')).toBeTruthy();

});

it('should make the name control required', () => {

let control = component.form.get('name');

control.setValue('');

expect(control.value).toBeFalsy();

});

});

### Event Emitter Testing

import { VoteComponent } from './vote.component';

describe('VoteComponent', () => {

var component: VoteComponent;

beforeEach(() => {

component = new VoteComponent();

});

it('should raise voteChanged event when upvoted', () => {

let totalVotes = null;

component.voteChanged.subscribe(tv=>{

totalVotes = tv;

});

component.upVote();

expect(totalVotes).not.toBeNull();

//more specific

expect(totalVotes).toBe(1);

});

});

import { EventEmitter } from '@angular/core';

export class VoteComponent {

totalVotes = 0;

voteChanged = new EventEmitter();

upVote() {

this.totalVotes++;

this.voteChanged.emit(this.totalVotes);

}

}

### Creating Service, Components with new keyword and Spying On them

import { TodosComponent, ToDoListModel } from './todos.component';

import { TodoService } from './todo.service';

import { of, empty, throwError, Observable } from 'rxjs';

import 'rxjs/add/observable/throw';

import 'rxjs/Observable'

describe('TodosComponent', () => {

let service: TodoService;

let component: TodosComponent;

beforeEach(() => {

service = new TodoService(null);

component = new TodosComponent(service);

});

it('should set todo properties returned by the service', () => {

let toDo : ToDoListModel = {id:0, body:"something" };

// control over the method, chanding the behaviour of the method.

/\*

\* much eaiser than callfake method so use, return value

\*/

//spyOn(service, 'getTodos').and.returnValue(of([{ id: 1, title: 'a' }]));

spyOn(service, 'getTodos').and.callFake(() => {

/\* here we need to stimulate the array as coming from the backend.

\* whenever we call the getToDos(), the function defined within callFake will be called,

\* when executing under the test environment

\*/

return of([toDo]);

});

component.ngOnInit();

expect(component.todos.length).toBeGreaterThan(0);

expect(component.todos).toContain(toDo);

});

it('should call the server AddService method',() => {

let spy = spyOn(service, 'add').and.callFake( t => {

return empty();

});

component.add();

expect(spy).toHaveBeenCalled();

});

it('should add a new todo',() => {

let toDo : ToDoListModel = {id:0, body:"something" };

let spy = spyOn(service, 'add').and.returnValue(

of([

toDo

])

);

component.add();

expect(component.todos.indexOf(toDo)).toBeGreaterThan(-1);

expect(component.todos.length).toBeGreaterThan(-1);

expect(spy).toHaveBeenCalled();

expect(spy).toHaveBeenCalledTimes(1);

});

xit('should set error message if server sends error message',() => {

let errorMessage = "there is a problem in network";

spyOn(service , 'add').and.returnValue(Observable.throw({status:404}));

component.add();

});

});

## Integration Testing

### Testbed

import { TestBed, ComponentFixture } from '@angular/core/testing';

import { VoterComponent } from './voter.component';

describe('VoterComponent', () => {

let component : VoterComponent;

let fixture : ComponentFixture<VoterComponent>;

beforeEach(() => {

/\*\*

\* In integration tests, we dont create component using new, new is for writing unit tests

\* In integration testing, we need to tell angular that components needs to be created

\*

\*/

TestBed.configureTestingModule({

declarations:[VoterComponent]

});

/\*\*

\* With component fixure we get the access to both, the component and its template

\*/

fixture = TestBed.createComponent(VoterComponent);

component = fixture.componentInstance;

/\*\*

\* root of the template

\*/

//fixture.nativeElement

/\*\*

\* It also has debugElement, which is the wrapper over nativeElement

\* this provides some usefull methods to query the html elements

\*/

//fixture.debugElement;

});

it('', () => {

});

});

import { async, ComponentFixture, TestBed } from '@angular/core/testing';

import { By } from '@angular/platform-browser';

import { HomeComponent } from './home.component';

import { HttpClientTestingModule } from '@angular/common/http/testing';

describe('Home Integration Testing', () => {

let component: HomeComponent;

let fixture: ComponentFixture<HomeComponent>;

// beforeEach(async(() => {

// TestBed.configureTestingModule({

// declarations: [GreeterIntegrationTestingComponent]

// })

// .compileComponents();

// }));

beforeEach(() => {

/\*\*

\* In the first beforeEach, we are telling angular to configure module

\* The components template is in a different file, we need to instruct angular to

\* compile the template as well as stylesheet with the component implementation.

\* the above compileComponents() so we have. All the components with the declaration array,

\* need to be compiled.

\*

\* As we are using webpack, we already have components compiled template in

\* one js file, so no need to explicitely load the files with asynch

\*/

TestBed.configureTestingModule({

declarations: [HomeComponent],

imports: [HttpClientTestingModule]

});

fixture = TestBed.createComponent(HomeComponent);

component = fixture.componentInstance;

component.totalVotes = 20;

fixture.detectChanges();

});

it('should create', () => {

expect(component).toBeTruthy();

});

it('should render total votes to 20', () => {

fixture.detectChanges();

let de = fixture.debugElement.query(By.css('.vote-count'));

/\*\* de : for debug element \*/

let el: HTMLElement = de.nativeElement;

let value = el.innerText;

expect(+value).toBe(20);

});

it('should apply class highlighted when user upvoted', () => {

fixture.detectChanges();

let de = fixture.debugElement.query(By.css('.vote-count'));

let el : HTMLElement = de.nativeElement;

expect(el.classList).toContain('highlighted');

});

it('should increase the total votes when clicking on upvote button', () => {

let button = fixture.debugElement.query(By.css('.upVoteClass'));

button.triggerEventHandler('click', null);

expect( component.totalVotes).toBe(21);

});

});

### With Async

xdescribe('TodosComponent', () => {

let component: TodosComponent;

let fixture: ComponentFixture<TodosComponent>;

beforeEach(async(() => {

TestBed.configureTestingModule({

declarations: [ TodosComponent ]

})

.compileComponents();

}));

beforeEach(() => {

fixture = TestBed.createComponent(TodosComponent);

component = fixture.componentInstance;

fixture.detectChanges();

});

it('should create', () => {

expect(component).toBeTruthy();

});

});

### Query **By**

import { async, ComponentFixture, TestBed } from '@angular/core/testing';

import { By } from '@angular/platform-browser';

import { HomeComponent } from './home.component';

import { HttpClientTestingModule } from '@angular/common/http/testing';

describe('Home Integration Testing', () => {

let component: HomeComponent;

let fixture: ComponentFixture<HomeComponent>;

// beforeEach(async(() => {

// TestBed.configureTestingModule({

// declarations: [GreeterIntegrationTestingComponent]

// })

// .compileComponents();

// }));

beforeEach(() => {

/\*\*

\* In the first beforeEach, we are telling angular to configure module

\* The components template is in a different file, we need to instruct angular to

\* compile the template as well as stylesheet with the component implementation.

\* the above compileComponents() so we have. All the components with the declaration array,

\* need to be compiled.

\*

\* As we are using webpack, we already have components compiled template in

\* one js file, so no need to explicitely load the files with asynch

\*/

TestBed.configureTestingModule({

declarations: [HomeComponent],

imports: [HttpClientTestingModule]

});

fixture = TestBed.createComponent(HomeComponent);

component = fixture.componentInstance;

component.totalVotes = 20;

fixture.detectChanges();

});

it('should create', () => {

expect(component).toBeTruthy();

});

it('should render total votes to 20', () => {

fixture.detectChanges();

let de = fixture.debugElement.query(By.css('.vote-count'));

/\*\* de : for debug element \*/

let el: HTMLElement = de.nativeElement;

let value = el.innerText;

expect(+value).toBe(20);

});

it('should apply class highlighted when user upvoted', () => {

fixture.detectChanges();

let de = fixture.debugElement.query(By.css('.vote-count'));

let el : HTMLElement = de.nativeElement;

expect(el.classList).toContain('highlighted');

});

it('should increase the total votes when clicking on upvote button', () => {

let button = fixture.debugElement.query(By.css('.upVoteClass'));

button.triggerEventHandler('click', null);

expect( component.totalVotes).toBe(21);

});

});

## spyOn

Changes the behavior of the method. When it is called, we can specify what to be returned by that method.

## Stub

Stub is nothing but, dummy implementation.

## Testing router

class RouterStub {

navigate(param){

}

}

class ActivatedRouteStub{

private subject = new Subject();

push(value){

this.subject.next(value);

}

get params(){

return this.subject.asObservable();

}

}

it('should navigate the user to not found page', () => {

let router = TestBed.get(Router);

let spy = spyOn(router, 'navigate');

let route : ActivatedRouteStub = TestBed.get(ActivatedRoute);

route.push({id : 0});

component.navigateToPostDetails();

expect(spy).toHaveBeenCalledWith(['/home'])

});

## Testing router outlet

1. it('should have a router outlet', () => {
2. const fixture = TestBed.createComponent(AppComponent);
3. const component = fixture.debugElement.componentInstance;
4. let element = fixture.debugElement.query(By.directive(RouterOutlet));
5. expect(element).not.toBeNull();
6. });

it('should have a link to Interpolation to way data binding', () => {

const fixture = TestBed.createComponent(AppComponent);

const component = fixture.debugElement.componentInstance;

fixture.detectChanges();

let elements = fixture.debugElement.queryAll(By.directive(RouterLinkWithHref));

let index = elements.findIndex(el => el.properties['href'] === "/interpolationand2waydatabinding");

//let index = elements.findIndex((e: DebugElement) => e.properties.href === "interpolationand2waydatabinding");

expect(index).toBeGreaterThan(-1);

});

## Attribute Directives Testing

@Component({

template:`

<input type="text" mydirective="someValue"

`

})

export class HostComponent {

}

describe("myDirective Testing", () => {

});

## Testing Custom and Structural Directives

@Component({

template: "<input type='text' id='m' appPhoneNumberFormat>"

})

class TestHoverFocusComponent {

}

describe('PhoneNumberFormatDirective Testing', () => {

let fixture: ComponentFixture<TestHoverFocusComponent>;

let component: TestHoverFocusComponent;

beforeEach(() => {

TestBed.configureTestingModule({

declarations: [TestHoverFocusComponent, PhoneNumberFormatDirective]

});

fixture = TestBed.createComponent(TestHoverFocusComponent);

component = fixture.componentInstance;

});

it('should format a number', () => {

let el = fixture.debugElement.query(By.css('input[id=m]'));

let s: HTMLInputElement = el.nativeElement;

s.value = "9096174175";

el.triggerEventHandler('keyup', null);

fixture.detectChanges();

expect(s.value).toBe('(+91) 9096-174-175');

});

});

# Animation (Web API Animations)

### fade

This is with style function provided in transition and animate function as a second parameter

|  |
| --- |
| @Component({  selector: 'app-to-do-list',  templateUrl: './to-do-list.component.html',  styleUrls: ['./to-do-list.component.css'],  animations:[    trigger('validationMessages',[     // state(),      transition('void => \*',[        style({          backgroundColor:'#F7F7F7', opacity: 0        }),        animate(2000, style({backgroundColor: 'white', opacity: 1}))      ]),      transition('\* => void',[        animate(500, style({opacity: 0}))      ])    ])  ]  }) |

### Animation with State function

|  |
| --- |
| @Component({  selector: 'app-to-do-list',  templateUrl: './to-do-list.component.html',  styleUrls: ['./to-do-list.component.css'],  animations:[    trigger('validationMessages',[      state('void', style({ opacity: 0})),      transition('void => \*',[        animate(2000)      ]),      transition('\* => void',[        animate(500)      ])    ])  ]  }) |

### Transition function with Repetition

animations:[

trigger('validationMessages',[

state('void', style({ opacity: 0})),

transition('void => \*',[

animate(2000)

]),

transition('\* => void',[

animate(500)

])

])

]

### Transition function repetition avoided with multiple state comma separated.

animations:[

trigger('validationMessages',[

state('void', style({ opacity: 0})),

transition('void => \*, \* => void',[

animate(2000)

])

])

### Bidirectional state in one go

animations:[

trigger('validationMessages',[

state('void', style({ opacity: 0})),

transition('void <=> \*',[

animate(2000)

])

])

]

### State aliases (:enter, :leave)

animations:[

trigger('validationMessages',[

state('void', style({ opacity: 0})),

transition(':enter, :leave',[

animate(2000)

])

])

]

### Fade in/out animation code

import { trigger, state, style, transition, animate } from "@angular/animations";

export let myFade = trigger('validationMessages',[

state('void', style({ opacity: 0})),

transition(':enter, :leave',[

animate(1000)

])

]);

export let removeSlide = trigger('removeSlide',[

transition(':leave',[

animate(700, style({ transform: 'translateX(-100px)' }))

])

]);

export let removeSlideWithEase = trigger('removeSlideWithEase',[

transition(':leave',[

animate('0.5s 0.5s ease-in', style({ transform: 'translateX(-100px)' }))

])

]);

export let removeSlideWithEase = trigger('removeSlideWithEase',[

transition(':leave',[

animate('0.5s cubic-bezier(.18,.63,.84,.31)', style({ transform: 'translateX(-100px)' }))

])

]);

### Keyframes

export let removeSlideWithEaseAndKeyFrame = trigger('removeSlideWithEaseAndKeyFrame',[

transition(':leave',[

animate('0.5s cubic-bezier(.18,.63,.84,.31)', keyframes([

style({

offset:.2,

opacity:1,

transform:'translateX(20px)'

}),

style({

offset:1,

opacity:0,

transform:'translateX(-100%)'

})

]))

])

]);

### Animation Events

<li class="list-group-item" \*ngFor="let task of tasks"

[@toDoAnimation]

(@toDoAnimation.start)="animationStarted($event)"

(@toDoAnimation.done)="animationDone($event)"

>

/\*\*

\* Animation things start here

\*/

animationStarted(e){

console.log('animation started');

}

animationDone(e){

console.log('animation done');

}

### Pseudo-Selectors Token

query(“:enter”),

query(“:leave”),

query(“:animating”),

query(“@trigger”),

query(“@\*”)

query(“:self”)

### Query elements and animate them

animations:[

trigger('todosAnimations',[

transition(":enter", [

query("h1",[

style({ transform: 'translateY(-20px)'}),

animate(1000)

])

])

]),

### Animating child elements

animations:[

trigger('todosAnimations',[

transition(":enter", [

query("h1",[

style({ transform: 'translateY(-20px)'}),

animate(1000)

]),

query('@toDoAnimation', [

animateChild()

])

])

]),

trigger('toDoAnimation', [

transition(':enter', [

useAnimation(fadeInAnimation, {

params: {

duration: '700ms',

easing : 'ease-out'

}

})

]),

transition(':leave', [

useAnimation(fadeOutAnimation, {

params: {

duration: '700ms',

easing : 'ease-in'

}

})

])

])

]

### Running Parallel Animations with Group

1. trigger('todosAnimations',[
2. transition(":enter", [
3. group([
4. query("h1",[
5. style({ transform: 'translateY(-20px)'}),
6. animate(1000)
7. ]),
8. query('@toDoAnimation', [
9. stagger("500ms", animateChild())
10. ])
11. ])
12. ])
13. ]),

### Custom State with Custom Panel

animations: [

trigger('expandCollapse', [

state("collapsed", style({

height: 0,

paddingTop: 0,

paddingBottom: 0,

opacity: 0

})),

state('expanded', style({

height: '\*',

padding: '\*',

overflow: 'auto'

})),

transition("collapsed => expanded", [

animate('500ms ease-out', style({

height: '\*',

paddingTop: '\*',

paddingBottom: '\*',

})),

animate("1s", style({

opacity : 1

}))

]),

transition("expanded => collapsed", animate('500ms ease-in'))

])

]

<div class="card">

<div class="card-header">

<ng-content select=".heading"></ng-content> <i class="fa float-right" style="cursor: pointer;"

[ngClass]="{'fa-caret-down' : !isExpanded, 'fa-caret-up' : isExpanded}" (click)="isExpanded=!isExpanded"></i>

</div>

<div class="card-body"

[@expandCollapse] ="isExpanded ? 'expanded' : 'collapsed'" > <ng-content select=".body"></ng-content> </div>

<!-- <div class="card-footer">Footer</div> -->

</div>

## Angular 7 Useful Data

|  |  |  |
| --- | --- | --- |
| **#** | **Item** | **Comments** |
| 1 | http://cubic-bezier.com/#.18,.63,.84,.31 | For creating different curves |
| 2 | http://bennettfeely.com/clippy/ | clipping |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |