1)

Unit Testing: -

Unit Testing is a type of software testing where individual units or components of a software are tested. The purpose is to validate that each unit of the software code performs as expected. Unit Testing is done during the development of an application by the developers.

Functional Testing: -

Functional Testing is a type of software testing that validates the software system against the functional requirements/specifications. The purpose of Functional tests is to test each function of the software application, by providing appropriate input, verifying the output against the Functional requirements.

2)

**1)For example if we wanted to test this function:**

function helloWorld() {

return 'Hello world!';

}

TEST CASE

describe('Hello world', () => { (1)

it('says hello', () => { (2)

expect(helloWorld()) (3)

.toEqual('Hello world!'); (4)

});});

**2)Testing Directives**

var sampleApp = AngularJS.module('sampleApp',[]);

sampleApp.directive('Arjun', function () {

return {

restrict: 'E',

replace: true,

template: '<h1>This is AngularJS Testing</h1>'

};

});

Test-Case

describe('Unit testing directives', function() {

var $compile,

$rootScope;

beforeEach(module('sampleApp'));

beforeEach(inject(function(\_$compile\_, \_$rootScope\_){

$compile = \_$compile\_;

$rootScope = \_$rootScope\_;

}));

it('Check the directive', function() {

// Compile a piece of HTML containing the directive

var element = $compile("<ng-Guru99></ng-Guru99>")($rootScope);

$rootScope.$digest();

expect(element.html()).toContain("This is AngularJS Testing");

});

});

**3)Testing for Exceptions**

function throwsError() {

throw new TypeError("A type error");}

-----------------------------------------------------------------

it('it should throw an exception', function () {

expect(throwsError).toThrow();});

**4)Testing @Input**

import { Component, Input, EventEmitter, Output, ViewChild, ElementRef } from '@angular/core';

@Component({

selector: 'title',

template: `

<p>{{ message }}</p>

<input #titleField type="text" />

<button (click)="handleButtonClick(titleField.value)">Change Title</button>

`,

styles: [`h1 { font-family: Lato; }`]

})

export class TitleComponent {

@Input() message: string; //1

@Output() changeTitleEvent:EventEmitter<string> = new EventEmitter(); //2

@ViewChild('titleField') titleField: ElementRef;

handleButtonClick(newTitle) {

if(newTitle) {

this.changeTitleEvent.emit(newTitle);

this.titleField.nativeElement.value = '';

}

}

}

**Test-Case**

// app.component.spec.ts

it('should correctly render the passed @Input value', () => {

component.message = 'Enter a new title'; // 1

fixture.detectChanges(); // 2

const compiled = fixture.debugElement.nativeElement; // 2

expect(compiled.querySelector('p').textContent).toBe('Enter a new title'); // 3

});

1. **Testing For Output**

**Test-Case for above code only**

const button = fixture.nativeElement.querySelector('button');

fixture.nativeElement.querySelector('input').value = 'A new title'; // 2

const inputText = fixture.nativeElement.querySelector('input').value;

button.click(); // 3

fixture.detectChanges();

expect(component.changeTitleEvent.emit).toHaveBeenCalledWith(inputText); // 4

});

1. **Testing Component**

import { Component, OnInit } from '@angular/core';

@Component({

selector: 'app-pizza',

templateUrl: './pizza.component.html',

styleUrls: ['./pizza.component.css']

})

export class PizzaComponent implements OnInit {

title = "I love pizza!"

constructor() { }

ngOnInit() {

}

}

**Test-Case**

it(`should have a title 'I love pizza!'`, async(() => {

fixture = TestBed.createComponent(PizzaComponent);

component = fixture.debugElement.componentInstance;

expect(component.title).toEqual('I love pizza!');

}));

## T**o test an async operation in Angular**

fetchQuotesFromServer() {

return new Promise((resolve, reject) => {

setTimeout(() => {

resolve([new QuoteModel("I love unit testing", "Mon 4, 2018")]);

}, 2000);

});

}

Test-Case

it("should fetch data asynchronously", async () => {

const fakedFetchedList = [

new QuoteModel("I love unit testing", "Mon 4, 2018")

];

const quoteService = fixture.debugElement.injector.get(QuoteService);

let spy = spyOn(quoteService, "fetchQuotesFromServer").and.returnValue(

Promise.resolve(fakedFetchedList)

);

fixture.detectChanges();

fixture.whenStable().then(() => {

expect(component.fetchedList).toBe(fakedFetchedList);

});

});

8)

describe('Adder', () => {

});

Test-Case

describe('Adder', () => {

// A jasmine spec

it('should be able to add two whole numbers', () => {

expect(Adder.add(2, 2)).toEqual(4);

});

it('should be able to add a whole number and a negative number', () => {

expect(Adder.add(2, -1)).toEqual(1);

});

it('should be able to add a whole number and a zero', () => {

expect(Adder.add(2, 0)).toEqual(2);

});});

**9)Dealing with timeouts**

const TIMEOUT\_DELAY = 250;

//...

it('should do something async', (done) => {

// \* arrange

const ob = { id: 1 };

// \* act

component.setSelectedAfterATimeout(ob);

// \* assert

setTimeout(() => {

expect(component.selected.id).toBe(ob.id);

done(); // let Jasmine know that you are done testing

}, TIMEOUT\_DELAY);

});

10)

@Component({

selector: 'app-todos',

template: `

<div \*ngFor="let todo of todos" class="todo">

{{todo.id}}

</div>

`

})

export class TodosComponent implements OnInit {

todos = [];

constructor(private todosService: TodosService) { }

ngOnInit() {

this.todosService.get().subscribe(todos => {

this.todos = todos;

});

}

}

Test-Case

const todosServiceStub = {

get() {

const todos = [{id: 1}];

return of( todos );

}

};

describe('TodosComponent', () => {

beforeEach(() => {

TestBed.configureTestingModule({

declarations: [ TodosComponent ],

providers: [{provide: TodosService, useValue: todosServiceStub}]

})

});

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

fixture.detectChanges();

expect(element.querySelectorAll('.todo').length).toEqual(1);

});

});