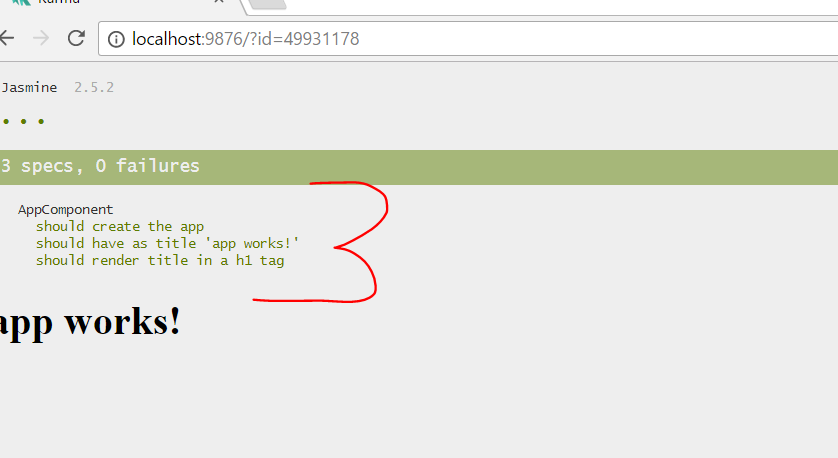
# Unit Testing help document

1. Create a new project using **ng new**
2. After successful creation of the project run the command **npm test**
3. It should open the browser and you will get test running as shown in the image below :



So 3 out of 3 tests are passed.

## Testing a pipe

Add a file called filesize.pipe.ts and create a pipe as shown in the listing below:

import { Pipe, PipeTransform } from '@angular/core';

@Pipe({

name: 'filesize'

})

export class FileSizePipe implements PipeTransform {

transform(size: number, extension: string = 'MB') {

return (size / (1024 \* 1024)).toFixed(2) + extension;

}

}

Write spec file to test pipe as shown below. File name should be **filesize.pipe.spec.ts**

import { FileSizePipe } from './filesize.pipe';

describe('FileSizePipe', () => {

const pipe = new FileSizePipe();

it('should convert bytes to megabyte', () => {

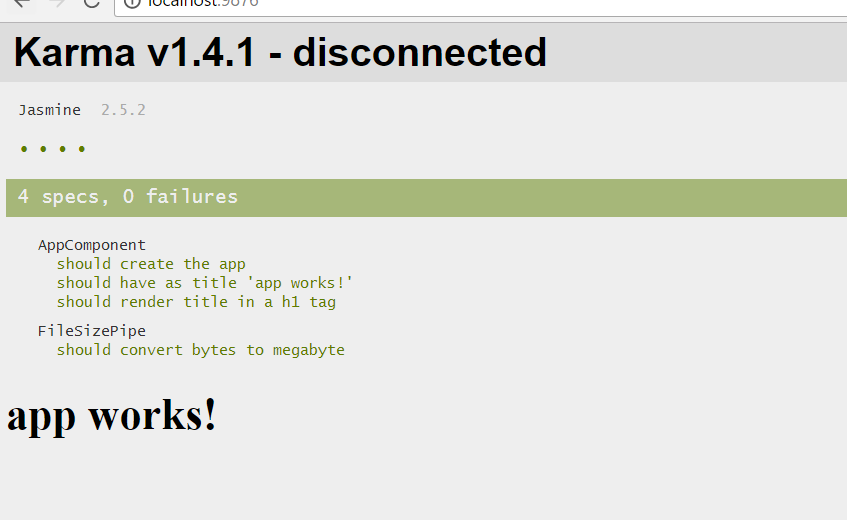
expect(pipe.transform(123456789)).toBe('117.74MB');

expect(pipe.transform(987654321)).toBe('941.90MB');

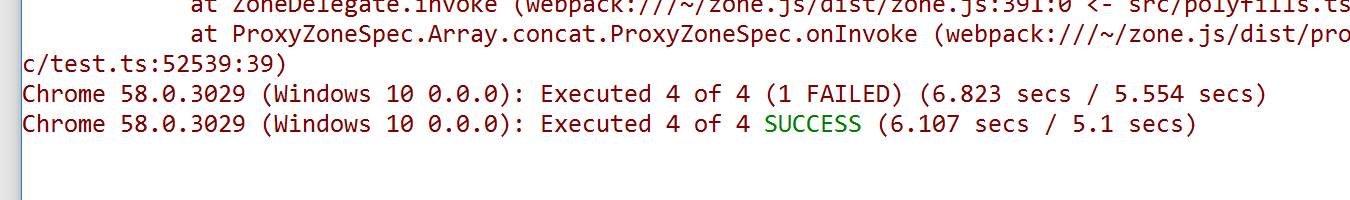
});

})

Save and run and all 4 unit test should pass as shown below:



And on the command prompt



### Testing Pipe inside Components or Shallow Pipe Testing

Modify filesize.pipe.spec.ts as below. Now we have two subscribe, one for the shallow pipe and one for isolate pipe.

import { FileSizePipe } from './filesize.pipe';

describe('FileSizePipe', () => {

describe('shallow filesize pipe test',()=>{

})

describe('isolate filesize pipe test', () => {

const pipe = new FileSizePipe();

it('should convert bytes to megabyte', () => {

expect(pipe.transform(123456789)).toBe('117.74MB');

expect(pipe.transform(987654321)).toBe('941.90MB');

});

})

})

Import these two statements.

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

import {BrowserDynamicTestingModule,platformBrowserDynamicTesting} from '@angular/platform-browser-dynamic/testing';

Next create the test environment like below. This is like bootstrapping angular modules.

import {BrowserDynamicTestingModule,platformBrowserDynamicTesting} from '@angular/platform-browser-dynamic/testing';

TestBed.initTestEnvironment(

BrowserDynamicTestingModule,

platformBrowserDynamicTesting()

);

import { FileSizePipe } from './filesize.pipe';

Next, we need to create a component to test the pipe. To do that we need to create a Componnet dynamically in the spec file. To do that first import Component as shown below:

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

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

Create the component like below. We are creating a dynamic components as below

describe('shallow filesize pipe test',()=>{

@Component({

template:`{{size | fileSize : suffix}}`

})

class TestComponent{

suffix;

size = 123456789;

}

})

Next create variable to refer the component and ComponentFixture. It is used to get reference of the component for change detection etc.

@Component({

template:`{{size | fileSize : suffix}}`

})

class TestComponent{

suffix;

size = 123456789;

}

let component : TestComponent;

let fixture : ComponentFixture<TestComponent>;

Do not forget to import ComponentFixture as below ,

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

Next we need to create test module before any test get executed. To do that first add a HTML element variable and in beforeEach create the test module as shown below . It is very much like creating ngModule.

let component : TestComponent;

let fixture : ComponentFixture<TestComponent>;

let el : HTMLElement;

beforeEach(()=>{

TestBed.configureTestingModule({

declarations:[

FileSizePipe,

TestComponent

]

})

})

Next we need to -

* create component
* get instance of component
* get native element

To do that redefine fixture, el and component variable as below

beforeEach(()=>{

TestBed.configureTestingModule({

declarations:[

FileSizePipe,

TestComponent

]

})

})

fixture = TestBed.createComponent(TestComponent);

component = fixture.componentInstance;

el = fixture.nativeElement;

})

Go and write the test as below

fixture = TestBed.createComponent(TestComponent);

component = fixture.componentInstance;

el = fixture.nativeElement;

it('should convert bytes to megabyte', () => {

fixture.detectChanges();

console.log(el);

});

})

Write test like this to test pipe in the component

it('should convert bytes to megabyte', () => {

fixture.detectChanges();

console.log(el);

expect(el.textContent).toContain('117.74MB');

});

Service Testing

In the project, add **product.ts** as shown in the listing below:

export interface Product {

id: number,

price: number,

name: string

}

export interface Item {

product\_id: number,

quantity: number

}

In the project, add **stock-inventory.service.ts** as below

import { Product, Item } from './product';

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

import { Http, Response } from '@angular/http';

import { Observable } from 'rxjs/Observable';

import 'rxjs/add/operator/map';

import 'rxjs/add/operator/catch';

import 'rxjs/add/observable/throw';

@Injectable()

export class StockInventoryService {

constructor(private http: Http) {}

getCartItems(): Observable<Item[]> {

return this.http

.get('/api/cart')

.map((response: Response) => response.json())

.catch((error: any) => Observable.throw(error.json()));

}

getProducts(): Observable<Product[]> {

return this.http

.get('/api/products')

.map((response: Response) => response.json())

.catch((error: any) => Observable.throw(error.json()));

}

}

We will test this service like below. **We are testing service with dependency** . Set up test as below : **stock-inventory.service.spec.ts**

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

import { BrowserDynamicTestingModule, platformBrowserDynamicTesting } from '@angular/platform-browser-dynamic/testing';

TestBed.initTestEnvironment(BrowserDynamicTestingModule,

platformBrowserDynamicTesting()

);

describe('servicetest',()=>{

beforeEach(()=>{

})

})

**Make sure to comment TestBed.initTestEnvironment because it is already done in test.ts**

To test service import statement as below:

import {StockInventoryService} from './stock-inventory.service';

Next in the BeforEach statement create the TestBed module as below

describe('servicetest', () => {

beforeEach(() => {

const bed = TestBed.configureTestingModule({

providers: [

StockInventoryService

]

})

})

})

We need to use **HTTP but we do not want to inject real HTTP, instead of that we will inject MockHttp** as shown below

class MockHttp{

get(){

return createResponse([]);

}

}

describe('servicetest',()=>{

Next, add createResponse function as shown below

function createResponse(body) {

return Observable.of(

new Response(new ResponseOptions({ body: JSON.stringify(body) }))

)

}

class MockHttp {

get() {

return createResponse([]);

}

}

Import the statements as below:

import { BrowserDynamicTestingModule, platformBrowserDynamicTesting } from '@angular/platform-browser-dynamic/testing';

import {Observable} from 'rxjs/Observable';

import 'rxjs/add/observable/of';

import {Response,ResponseOptions} from '@angular/http';

Add two fake response item as shown below. We will use them with MockHttp to mock out the response.

class MockHttp {

get() {

return createResponse([]);

}

}

const cartItems = [{ product\_id: 1, quanity: 10 }, { product\_id: 2, quanity: 12 }];

const productItems = [{ id: 1, price: 10, name: 'Test' }, { id: 1, price: 100, name: 'Test 1' }];

We need to inject MockHttp class. To do that import HTTP as below:

import {Http,Response,ResponseOptions} from '@angular/http';

Modify before each statement as below. Essentially, we are saying use MockHttp class instead of the HTTP class.

beforeEach(() => {

const bed = TestBed.configureTestingModule({

providers: [

StockInventoryService,

{ provide: Http, useClass: MockHttp }

]

});

})

Next, create some variables for service and Http. To do that modify code as shown below:

describe('servicetest',()=>{

let service : StockInventoryService;

let http : Http;

beforeEach(()=>{

const bed = TestBed.configureTestingModule({

providers:[

StockInventoryService,

{provide:Http,useClass:MockHttp}

]

});

http = bed.get(Http);

service = bed.get(StockInventoryService);

})

})

At this point, our testing environment has been set up and we need to work with writing test now.

beforeEach(() => {

const bed = TestBed.configureTestingModule({

providers: [

StockInventoryService,

{ provide: Http, useClass: MockHttp }

]

});

http = bed.get(Http);

service = bed.get(StockInventoryService);

});

it('should get cart items', () => {

spyOn(http, 'get').and.returnValue(createResponse([...cartItems]));

service.getCartItems().subscribe((result) => {

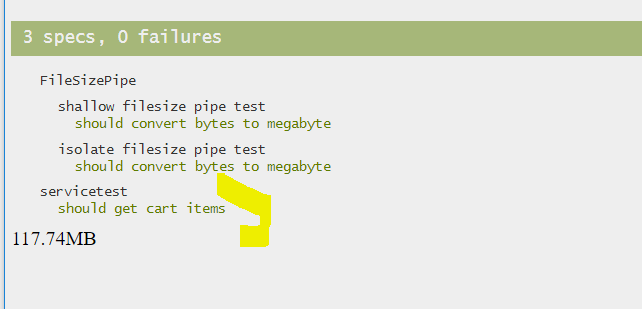
expect(result.length).toBe(2);

expect(result).toEqual(cartItems);

});

});

This will pass the test. As shown below :



Test the product items as below:

it('should get ptroduct items', () => {

spyOn(http, 'get').and.returnValue(createResponse([...productItems]));

service.getProducts().subscribe((result) => {

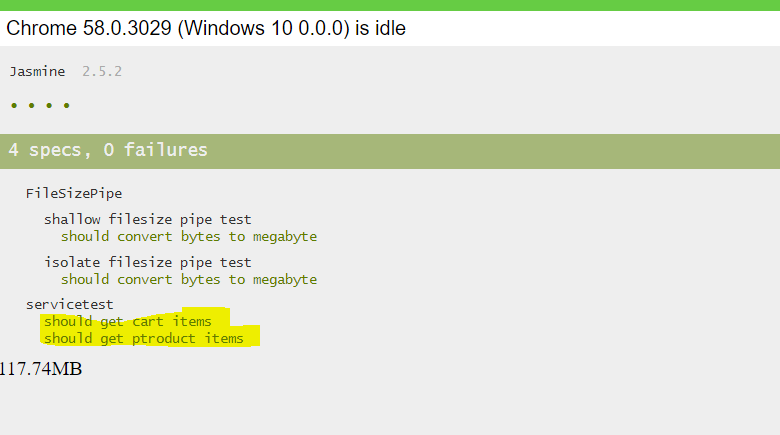
expect(result.length).toBe(2);

expect(result).toEqual(productItems);

});

});

Run the application and you will have test as below:



# Testing Components

Create a file called **stock-counter.component.spec.ts**

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

import { BrowserDynamicTestingModule, platformBrowserDynamicTesting } from '@angular/platform-browser-dynamic/testing';

import {StockCounterComponent} from './stock-counter.component';

Create a component class as below :

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

@Component({

})

export class StockCounterComponent{

}

Initialize the Test environment like below

TestBed.initTestEnvironment(BrowserDynamicTestingModule, platformBrowserDynamicTesting());

**Note: comment this line as Test env is already initialized in the test.ts**

Create a describe statement as shown below in the file.

describe('StockCounterComponent', () => {

});

Below the describe create initialize the testing module as shown below:

describe('StockCounterComponent', () => {

beforeEach(() => {

TestBed.configureTestingModule({

declarations: [StockCounterComponent]

});

});

});

Create the variable for fixture and component as shown below:

let component : StockCounterComponent;

let fixture : ComponentFixture<StockCounterComponent>;

beforeEach(()=>{

TestBed.configureTestingModule({

declarations:[StockCounterComponent]

});

});

Override them in before each as shown below . We need to create instance of StcokCounterComponent and assign its value to 0 as shown below:

beforeEach(()=>{

TestBed.configureTestingModule({

declarations:[StockCounterComponent]

});

fixture = TestBed.createComponent(StockCounterComponent);

component = fixture.componentInstance;

component.value = 0 ;

Run test to Unit test increment as below:

beforeEach(() => {

TestBed.configureTestingModule({

declarations: [StockCounterComponent]

});

fixture = TestBed.createComponent(StockCounterComponent);

component = fixture.componentInstance;

component.value = 0;

});

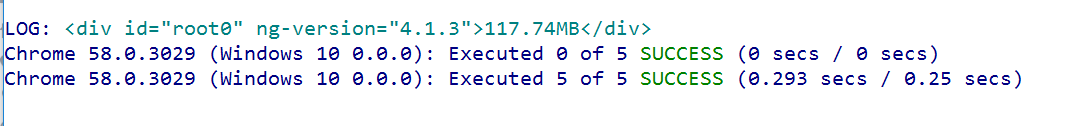
it('should increment correctly', () => {

component.increment();

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

})

At this point run the test and you should get all test passed as below:



After that, write other test as below:

it('should decrement correctly', () => {

component.increment()

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

component.decrement()

expect(component.value).toBe(0);

});

it('should not decrement below the minimum value', () => {

component.increment()

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

component.decrement()

expect(component.value).toBe(0);

component.decrement()

expect(component.value).toBe(0);

});

it('should not increment below the maximum value', () => {

for (let i = 0; i < 200; i++) {

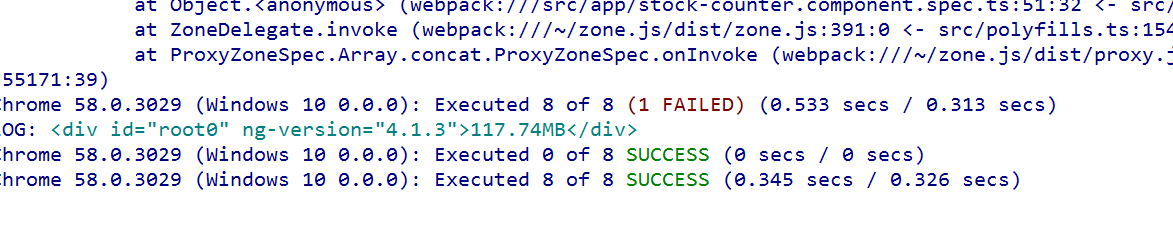
component.increment();

}

expect(component.value).toBe(100);

});

You will get all test running as shown below:



Create a Login Component - login.component.ts

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

export class User {

constructor(public email: string, public password: string) {

}

}

@Component({

selector: 'app-login',

template: `

<form>

<label>Email</label>

<input type="email"

#email>

<label>Password</label>

<input type="password"

#password>

<button type="button"

(click)="login(email.value, password.value)"

[disabled]="!enabled">Login

</button>

</form>

`

})

export class LoginComponent {

@Output() loggedIn = new EventEmitter<User>();

@Input() enabled = true;

login(email, password) {

console.log(`Login ${email} ${password}`);

if (email && password) {

console.log(`Emitting`);

this.loggedIn.emit(new User(email, password));

}

}

}

Type Test as below

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

import { BrowserDynamicTestingModule, platformBrowserDynamicTesting } from '@angular/platform-browser-dynamic/testing';

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

import { LoginComponent , User } from './login.component';

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

TestBed.initTestEnvironment(BrowserDynamicTestingModule,

platformBrowserDynamicTesting()

);

describe('Component: Login', () => {

let component: LoginComponent;

let fixture: ComponentFixture<LoginComponent>;

let submitEl: DebugElement;

let loginEl: DebugElement;

let passwordEl: DebugElement;

beforeEach(() => {

TestBed.configureTestingModule({

declarations: [LoginComponent]

});

// create component and test fixture

fixture = TestBed.createComponent(LoginComponent);

// get test component from the fixture

component = fixture.componentInstance;

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

loginEl = fixture.debugElement.query(By.css('input[type=email]'));

passwordEl = fixture.debugElement.query(By.css('input[type=password]'));

});

it('Setting enabled to false disables the submit button', () => {

component.enabled = false;

fixture.detectChanges();

expect(submitEl.nativeElement.disabled).toBeTruthy();

});

it('Entering email and password emits loggedIn event', () => {

let user: User;

loginEl.nativeElement.value = "test@example.com";

passwordEl.nativeElement.value = "12347";

component.loggedIn.subscribe((value) => user = value);

submitEl.triggerEventHandler('click', null);

expect(user.email).toBe("test@example.com");

expect(user.password).toBe("123456");

});

});