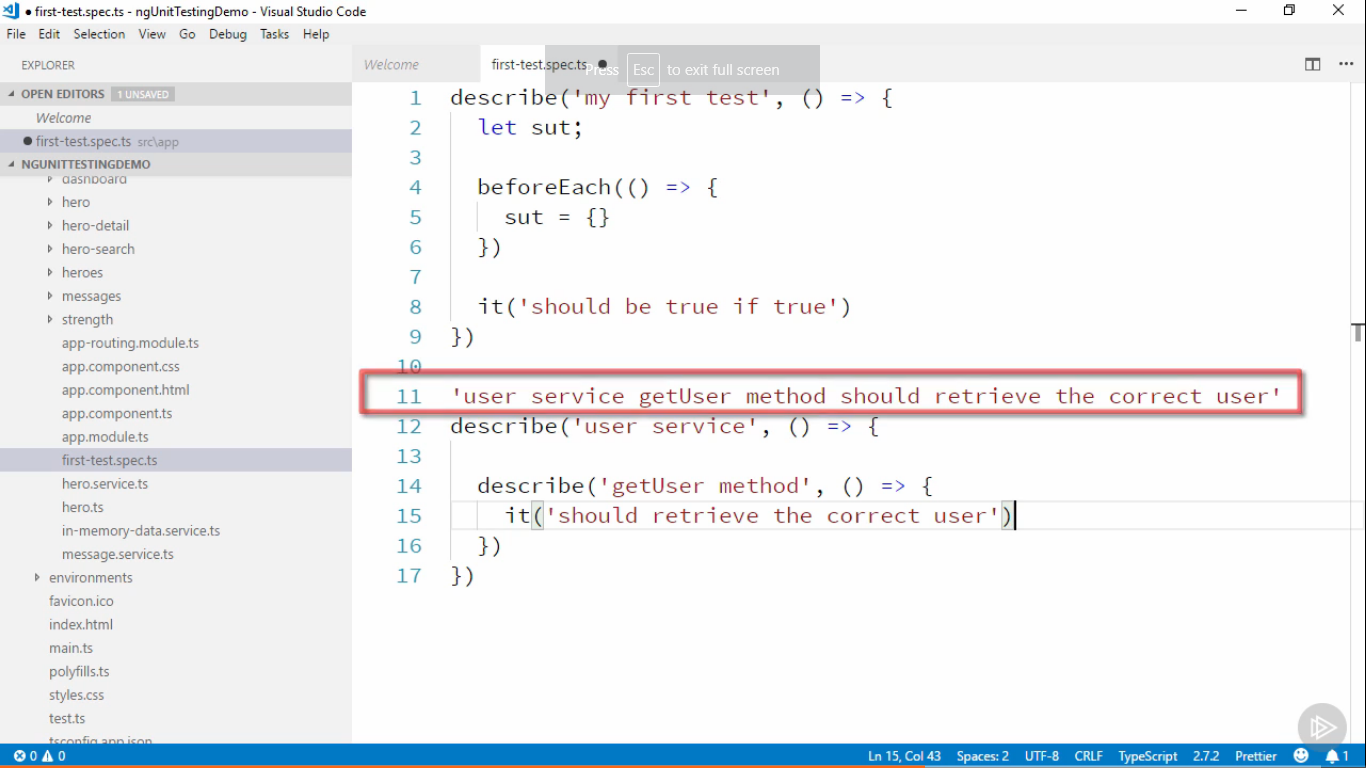
**This is jasmine test**

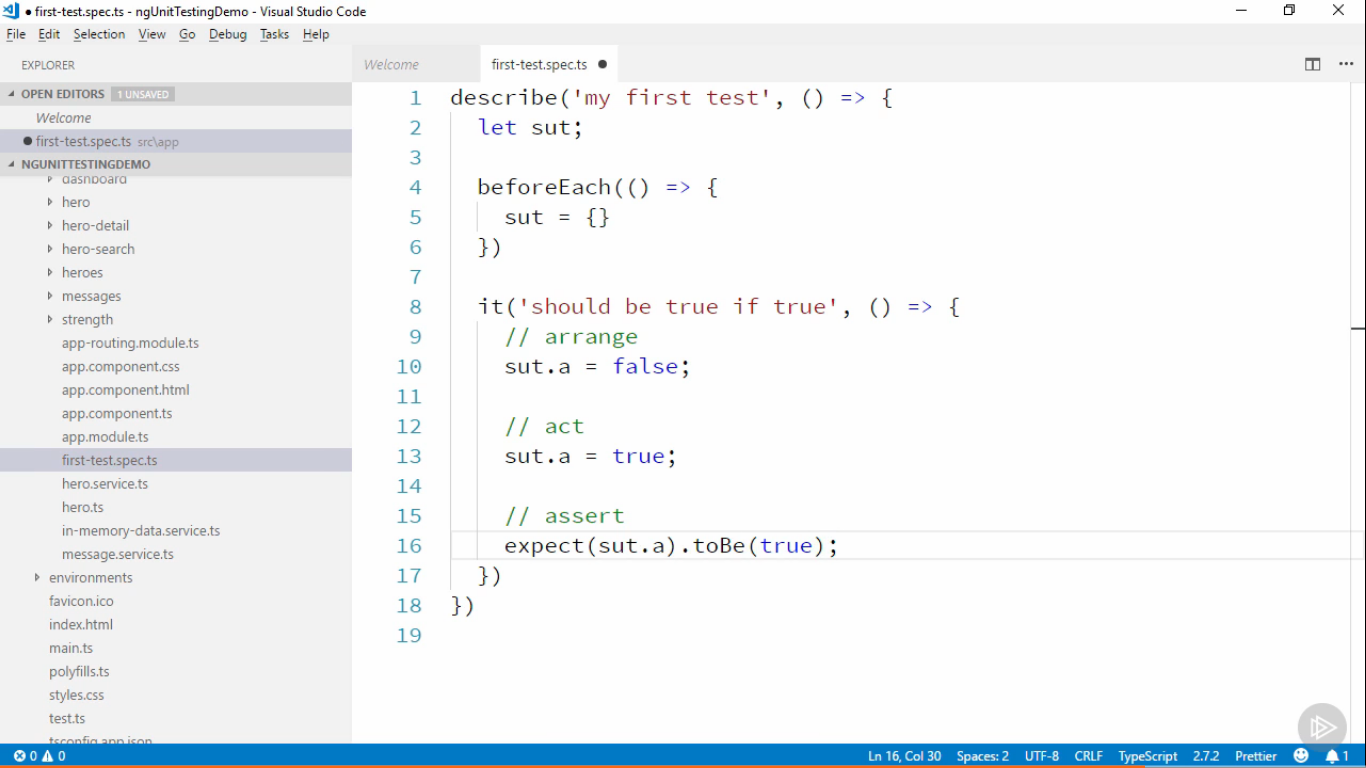
Describe – to combine the tests (sut stands for system under test)

beforeEach – this is used to set variables declared to empty because we don’t want to pollute the future tests with results of previous tests

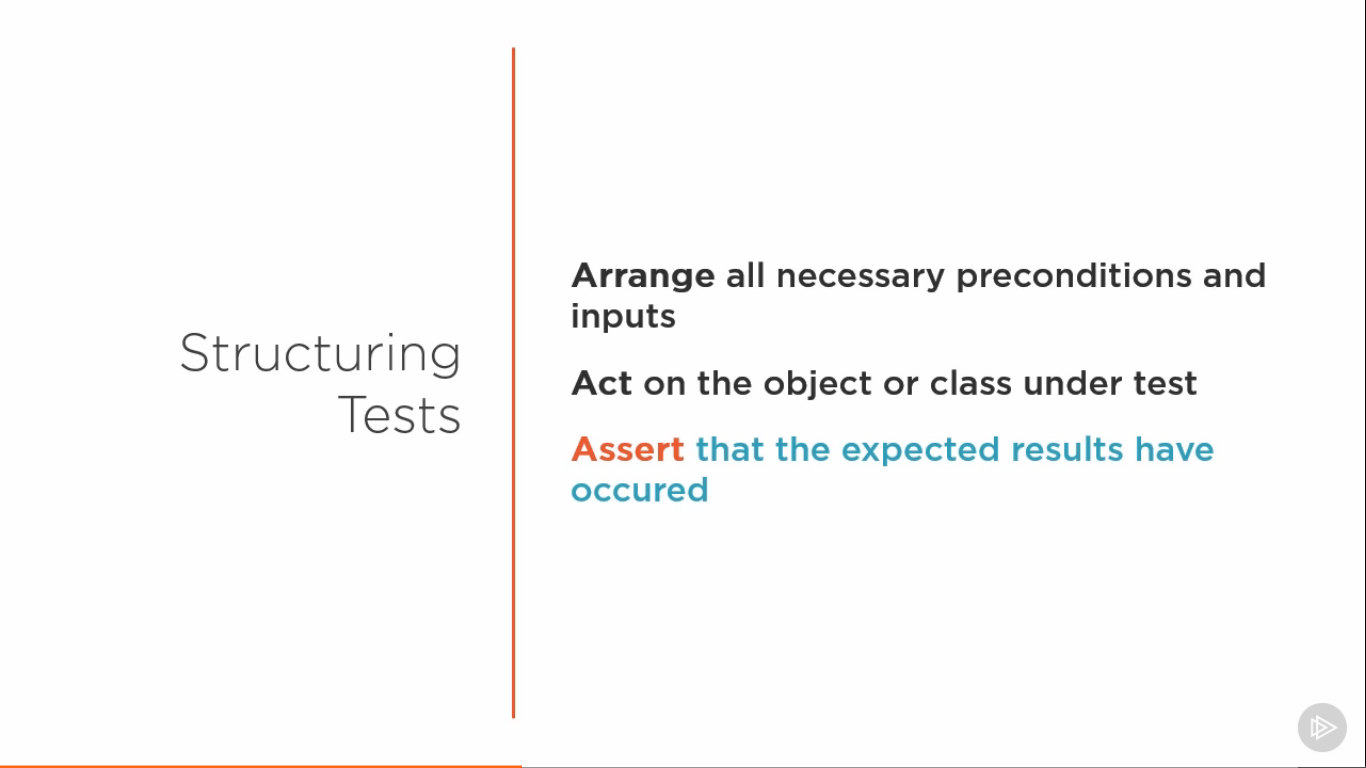
it – this where the test is written and it always have what the test is about or **‘should’**



AAA test structure



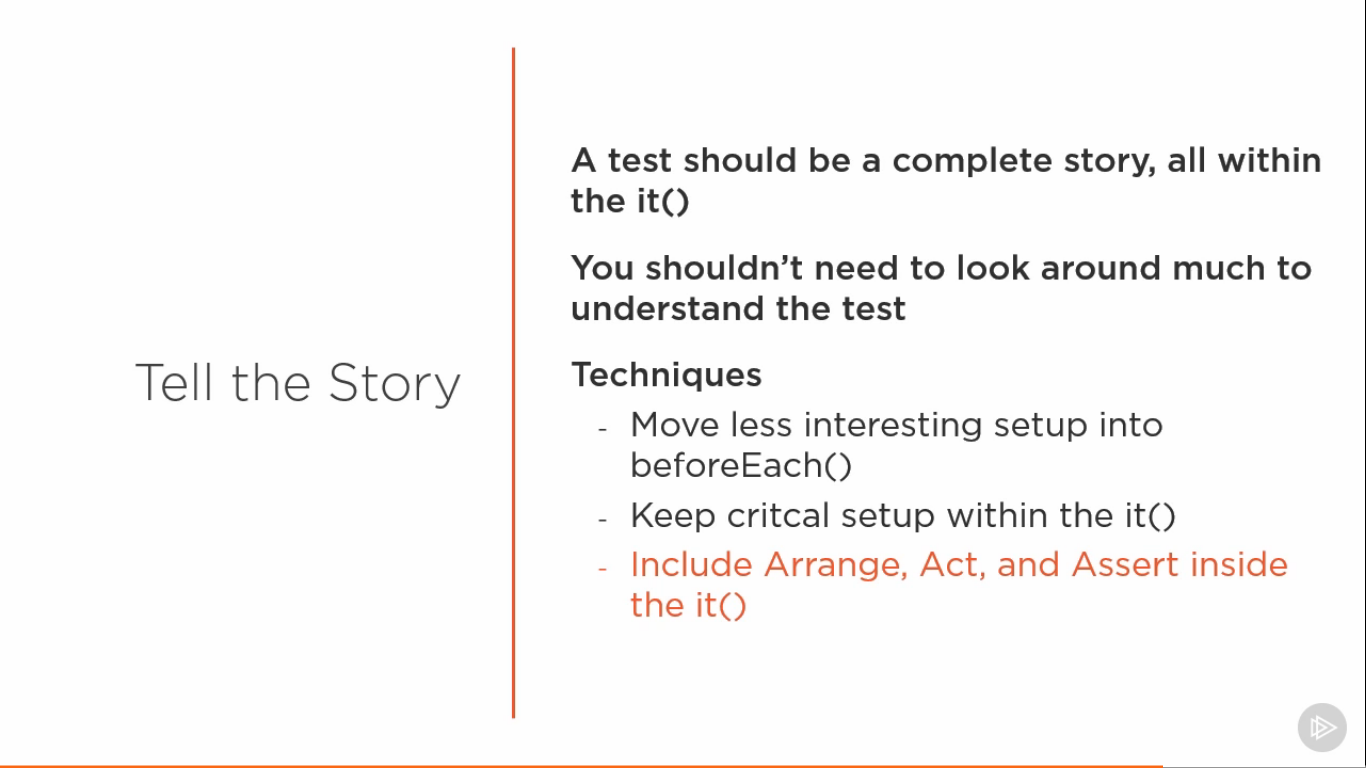
**Writing Good Unit Tests**



Concept in Programming



Good Practices



**Isolated Unit Tests**



Interaction test component

import { HeroesComponent } from "./heroes.component"

import { of } from "rxjs";

describe('Heroscomponent',()=>

{

    let component : HeroesComponent;

    let HEROES;

    let mockHeroSer;

    beforeEach(()=>{

        HEROES =[

            {id:1,name:'yes',strength:8},

            {id:2,name:'true',strength:5},

            {id:3,name:'false',strength:11}

        ]

    mockHeroSer = jasmine.createSpyObj(['delete1Hero'])

    component = new HeroesComponent(mockHeroSer)

})

describe('delete',()=>

{

    it('should remove',()=>{

        mockHeroSer.delete1Hero.and.returnValue(of(true))

        component.heroes = HEROES

        component.delete(HEROES[2])

        expect(component.heroes.length).toBe(2)

    })

    it('should remove with parameter',()=>{

        mockHeroSer.delete1Hero.and.returnValue(of(true))

        component.heroes = HEROES

        component.delete(HEROES[2])

        expect(mockHeroSer.delete1Hero).toHaveBeenCalledWith(HEROES[2])

    })

})

})

mockHero.Ser.delete1Hero.and.returnValue(of(true)) – this is used because an service returns an observable . We mock observable by writing this

**SHALLOW TESTS**

For zone and karma conflict issue use

"test": "ng test --source-map=false",

**TestBed** allows us test both component and its template running together

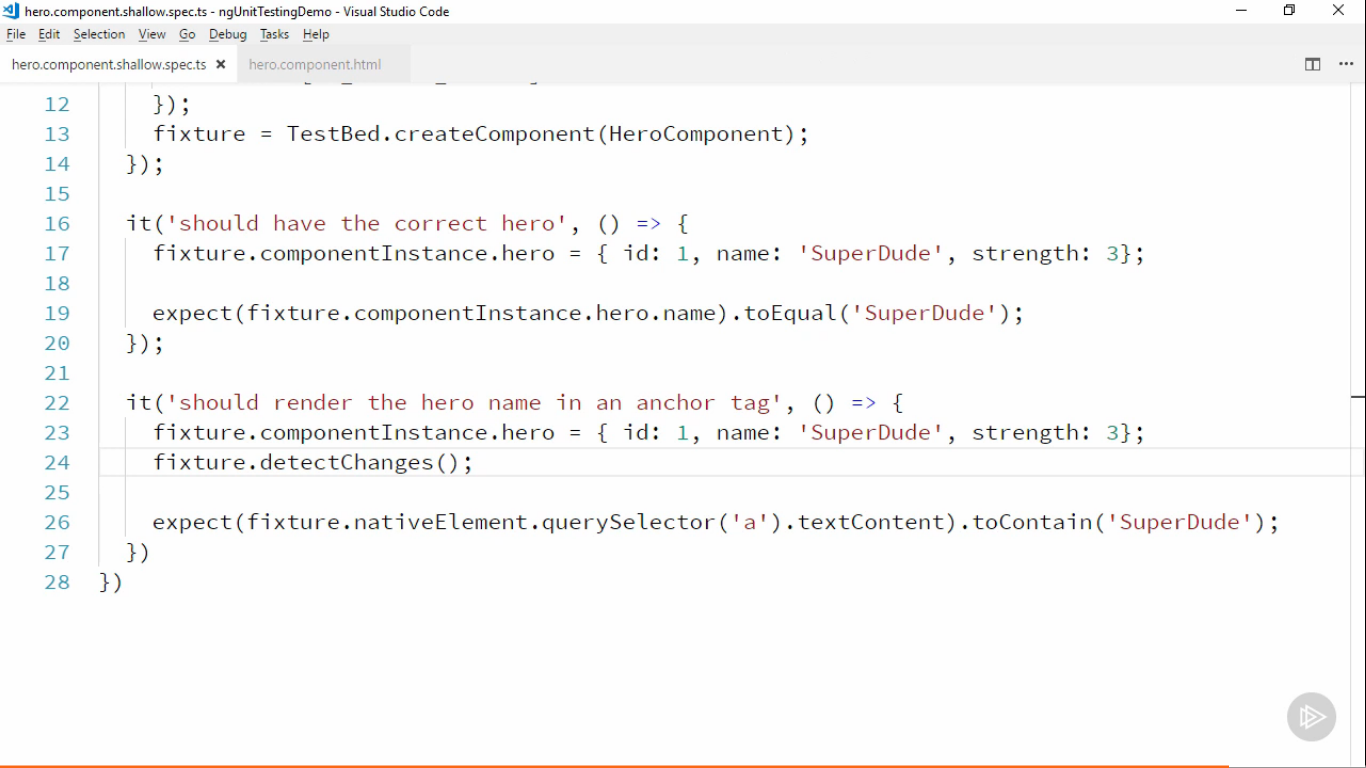
Using TestBed we create a module in which we add the component and to create it we use **create component method** and it returns a fixture

**Fixture** it is like a wrapper around the component. It has more properties than the component itself.

NO\_\_ERROR\_SCHEMA blocks all unknown element or error and allows the test to run. We should use **this only when you want to ignore errors** **that have no impact**. Html syntax errors are ignored .

Bindings are not updated until change detection runs.

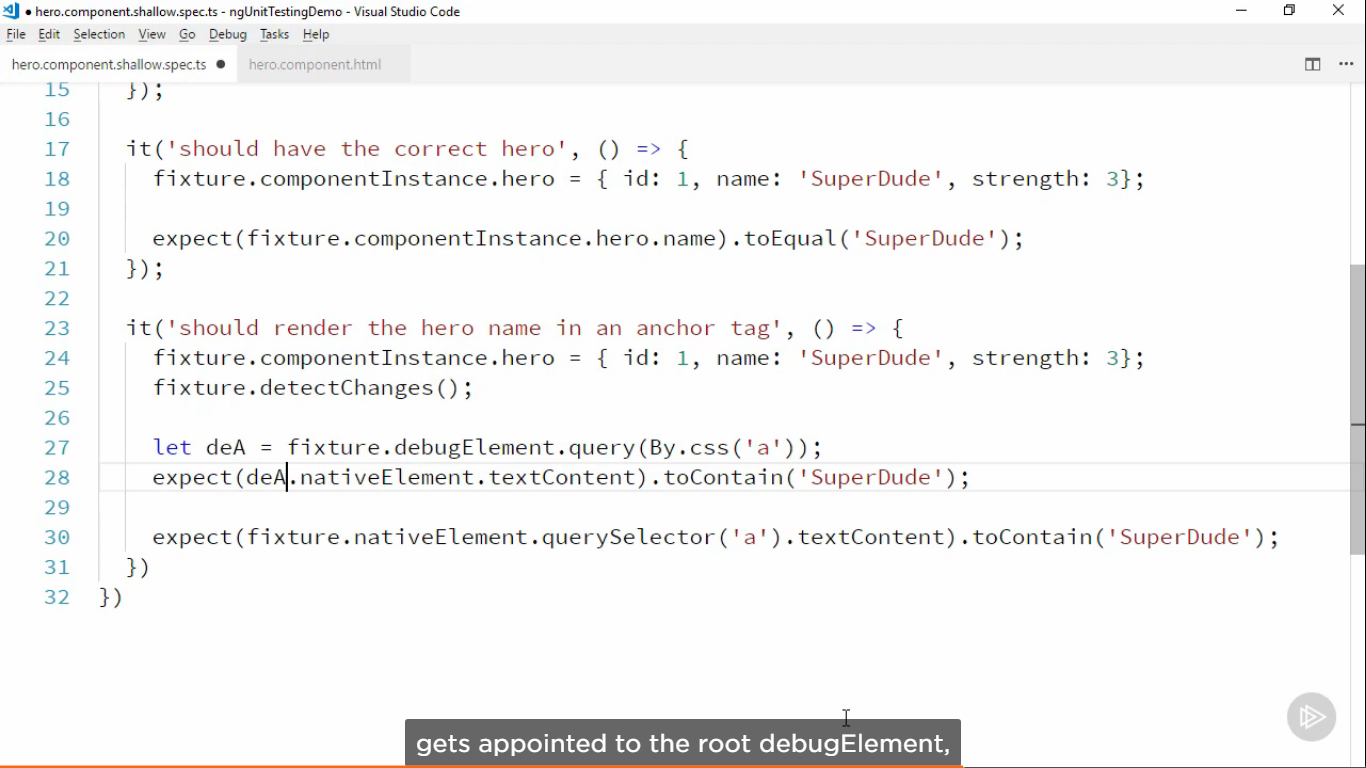
**In order for ngOnIt to fire we have to run change detection. In angular change detection causes lifecycle events to run. We can fire this by calling Fixture.detectChanges();**



Using the fixture we can access the DOM elements using the native Element method in which it has query selector which will have the elements (exposes dom api)

Debug Element access the root element of template, it is like a wrapper around dom node

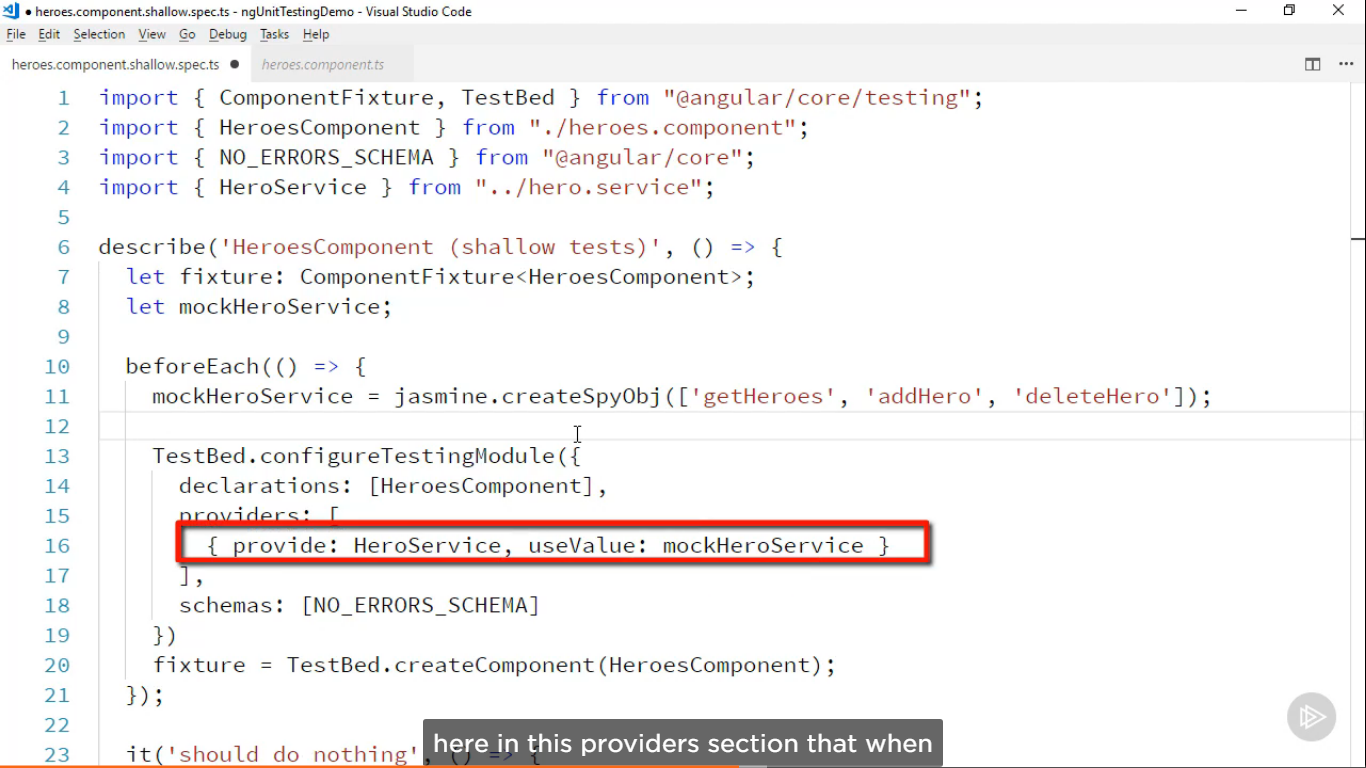
DebugElement has more functionality than nativeElement



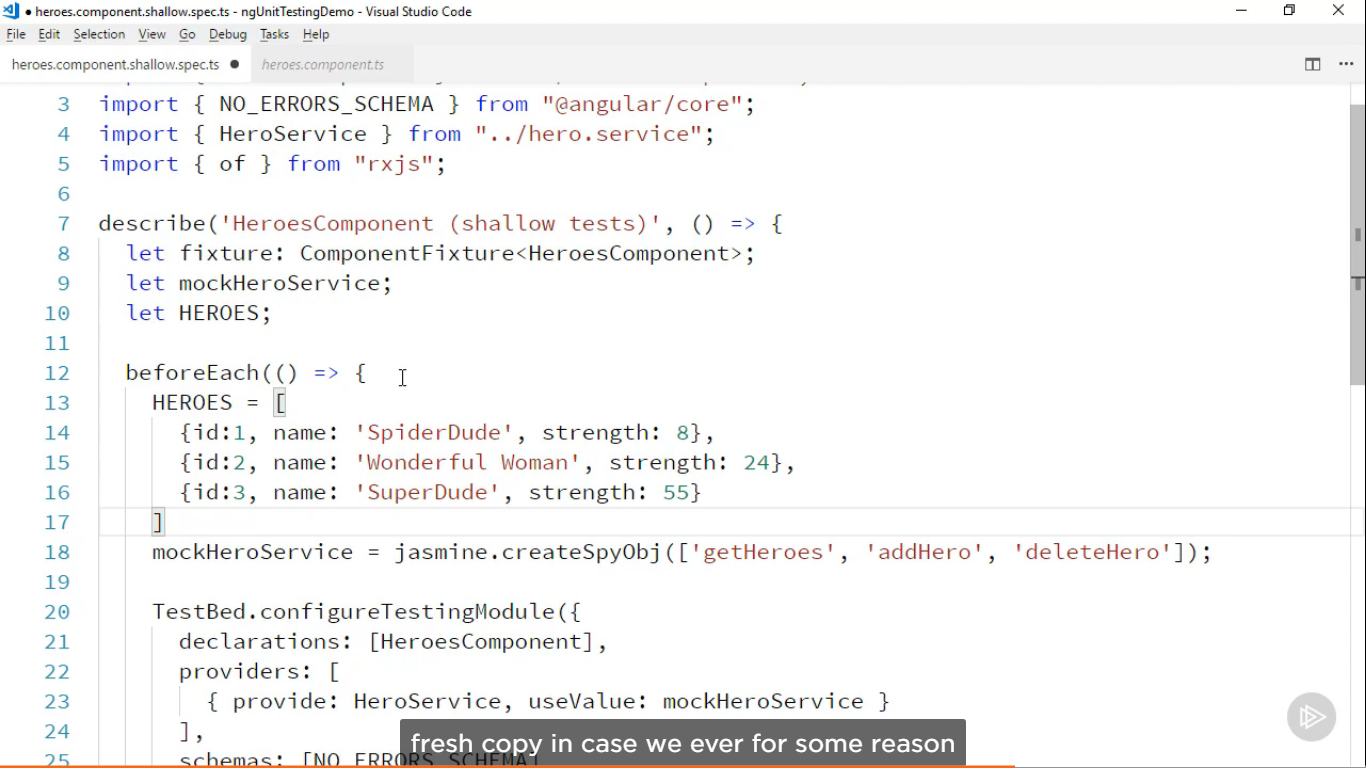
**More Shallow integration tests**

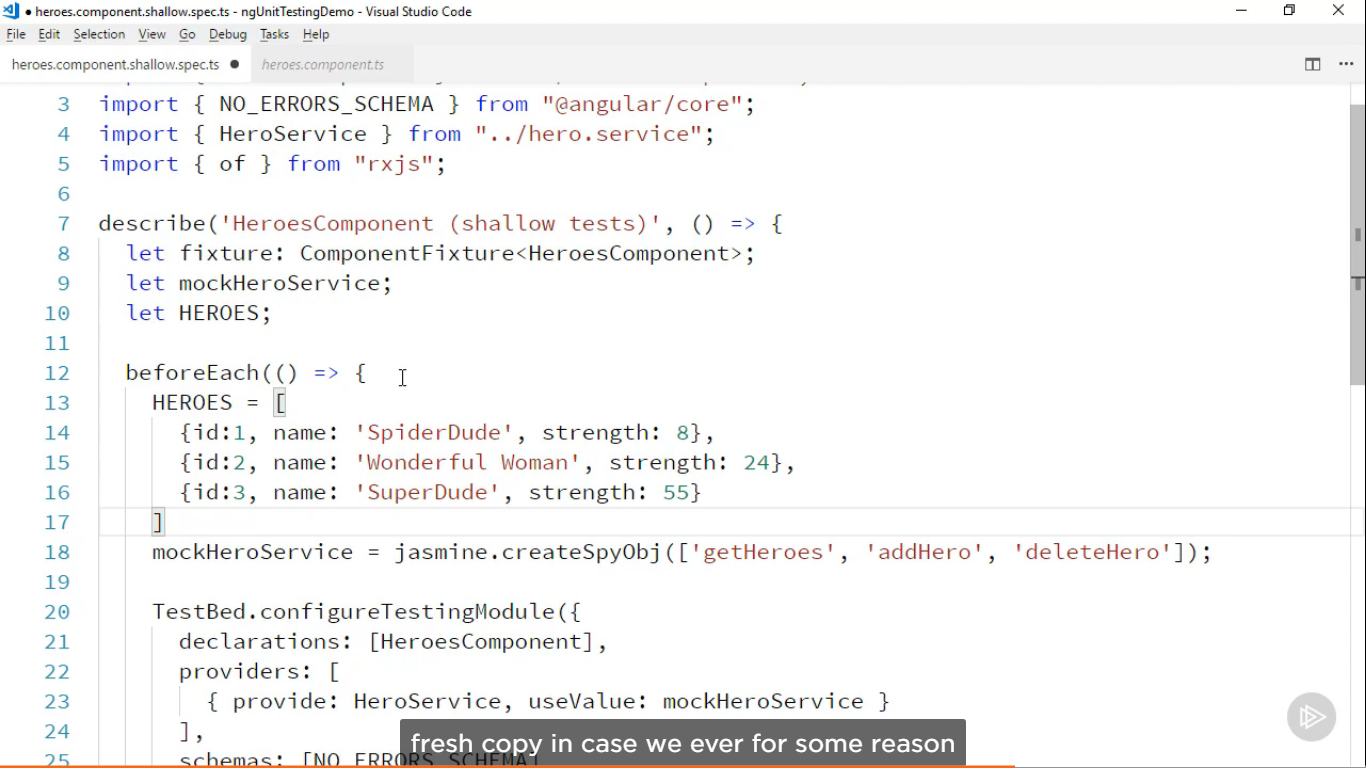
1) Mocking injected service

Writing the service in long hand way so to mock it



Testing get Hero method







2) Mocking Child Components

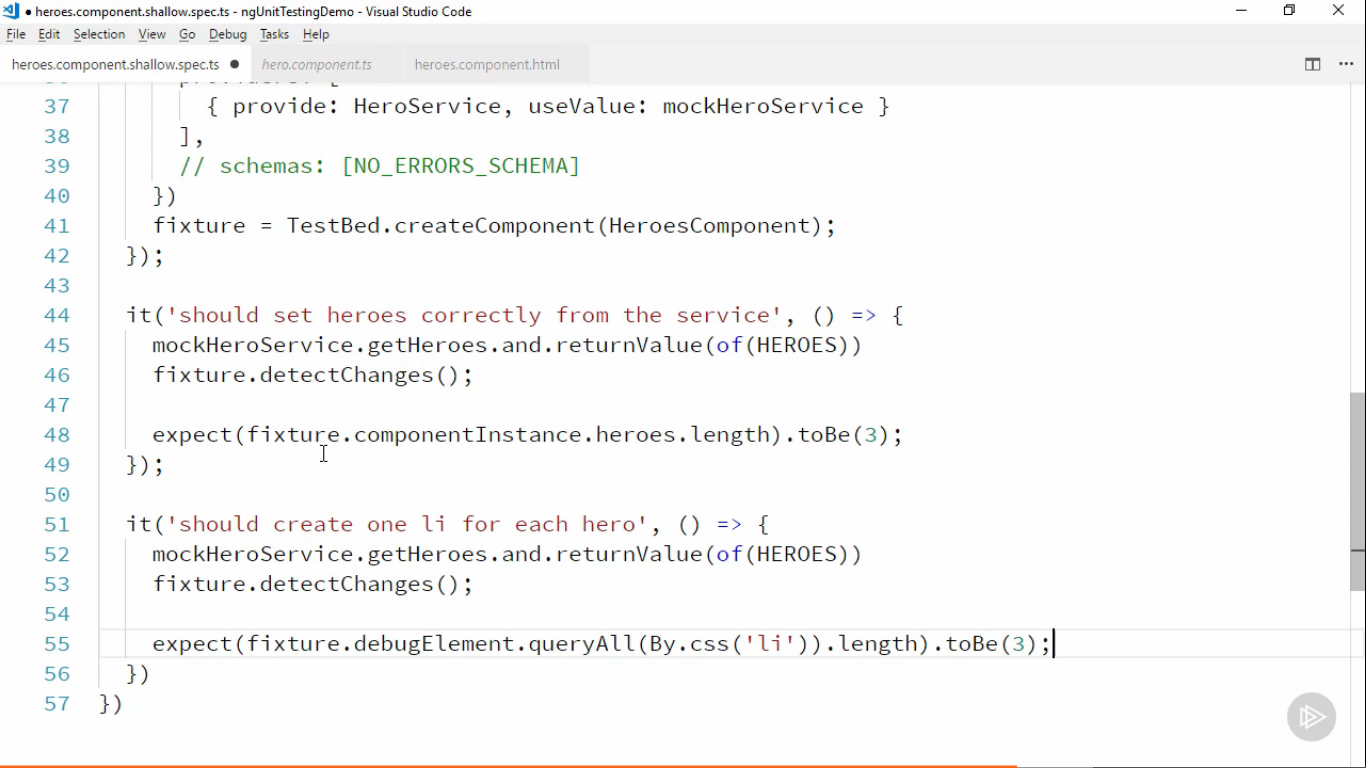
We can do this by writing a mock child component syntax above beforeEach. The syntax should be like this



And Add it to the testing module in declarations section

3) Dealing with List of elements

We will be counting the number of DOM elements created



**Deep Integration Tests**

In this we check with the live children component

