## Angular Test Bed

The Angular Test Bed (ATB) is a higher level *Angular Only* testing framework that allows us to easily test behaviours that depend on the Angular Framework.

We still write our tests in Jasmine and run using Karma but we now have a slightly easier way to create components, handle injection, test asynchronous behaviour and interact with our application.

**Learning Objectives**

* What is the ATB and how to use it.
* When to use ATB vs. plain vanilla Jasmine tests.

**Configuring**

Lets demonstrate how to use the *ATB* by converting the component we tested with plain vanilla Jasmine to one that uses the *ATB*.

/\* tslint:disable:no-unused-variable \*/

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

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

import {AuthService} from "./auth.service";

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

beforeEach(() => {

TestBed.configureTestingModule({

declarations: [LoginComponent],

providers: [AuthService]

});

});

})

;

We are importing TestBed and ComponentFixture from the Angular Testing module

In the beforeEach function for our test suite we *configure* a testing module using the TestBed class.

This creates a test *Angular Module* which we can use to instantiate components, perform dependency injection and so on.

(*Angular Module* will instantiate components, perform dependency injection and so on)

We configure it in exactly the same way as we would configure a normal NgModule. On this case we pass in the LoginComponent in the declarations and the AuthService in the providers.

**Fixtures and DI**

Once the ATB is setup we can then use it to instantiate components and resolve dependencies, like so:

/\* tslint:disable:no-unused-variable \*/

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

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

import {AuthService} from "./auth.service";

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

let component: LoginComponent;

let fixture: ComponentFixture<LoginComponent>;

①

let authService: AuthService;

beforeEach(() => {

TestBed.configureTestingModule({

declarations: [LoginComponent],

providers: [AuthService]

});

// create component and test fixture

fixture = TestBed.createComponent(LoginComponent);

②

// get test component from the fixture

component = fixture.componentInstance;

③

// UserService provided to the TestBed

authService = TestBed.get(AuthService);

④

});

})

;

1. A fixture is a wrapper for a component *and* it’s template.
2. We create an instance of a component fixture through the TestBed, this injects the AuthService into the component constructor. By creating the component through TestBed then it automatically injects all the services into the component constructor.
3. We can find the actual *component* from the componentInstance on the fixture.
4. We can get resolve dependencies using the TestBed injector by using the get function.

 Since the LoginComponent doesn’t have it’s own child injector the AuthService that

the gets injected in is the same one as we get from TestBed above.

**Test specs**

Now we’ve configured the TestBed and extracted the component and service we can run through the same test specs as before:

it('canLogin returns false when the user is not authenticated', () => {

spyOn(authService, 'isAuthenticated').and.returnValue(false);

expect(component.needsLogin()).toBeTruthy();

expect(authService.isAuthenticated).toHaveBeenCalled();

});

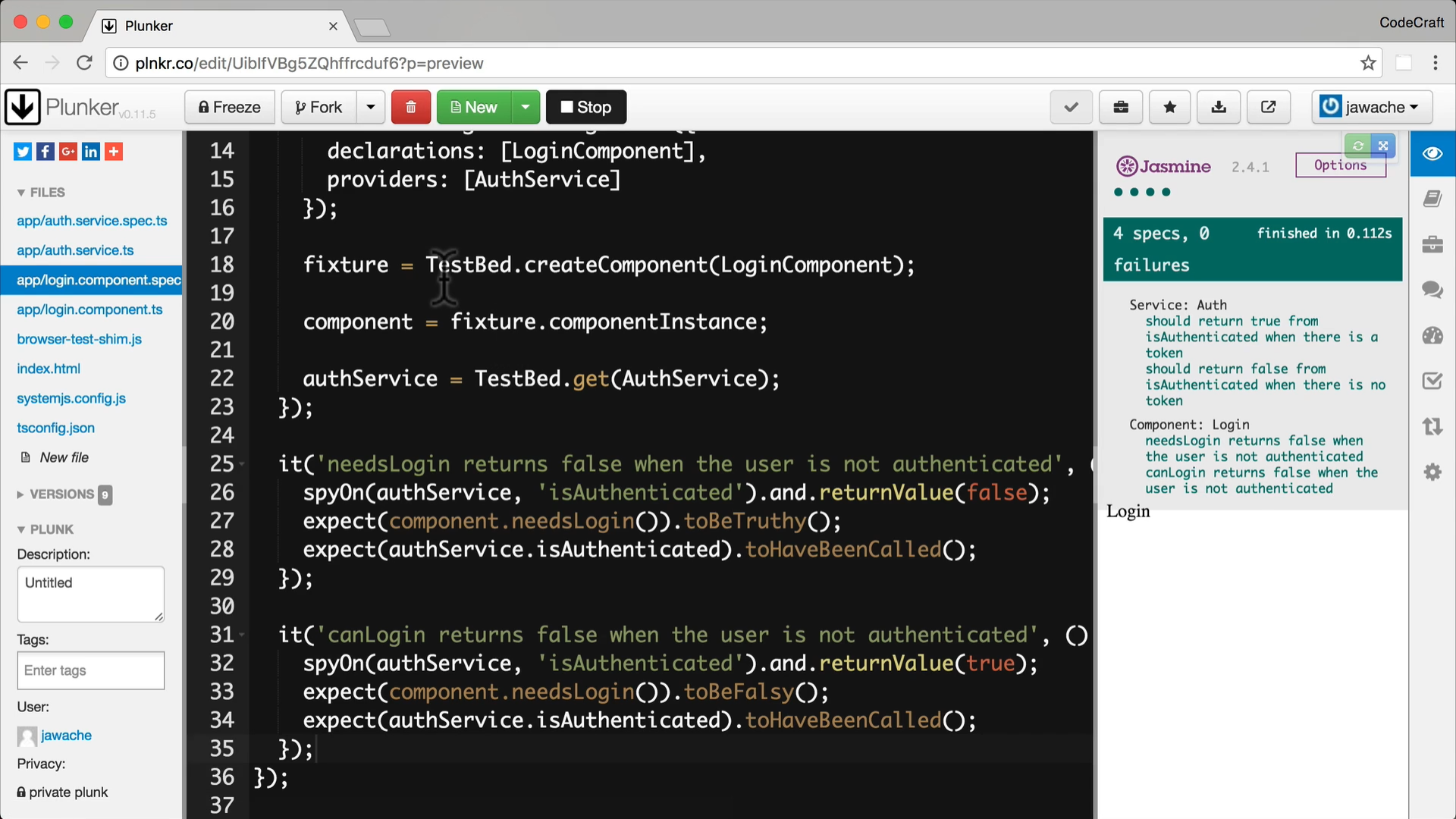
it('canLogin returns false when the user is not authenticated', () => {

spyOn(authService, 'isAuthenticated').and.returnValue(true);

expect(component.needsLogin()).toBeFalsy();

expect(authService.isAuthenticated).toHaveBeenCalled();

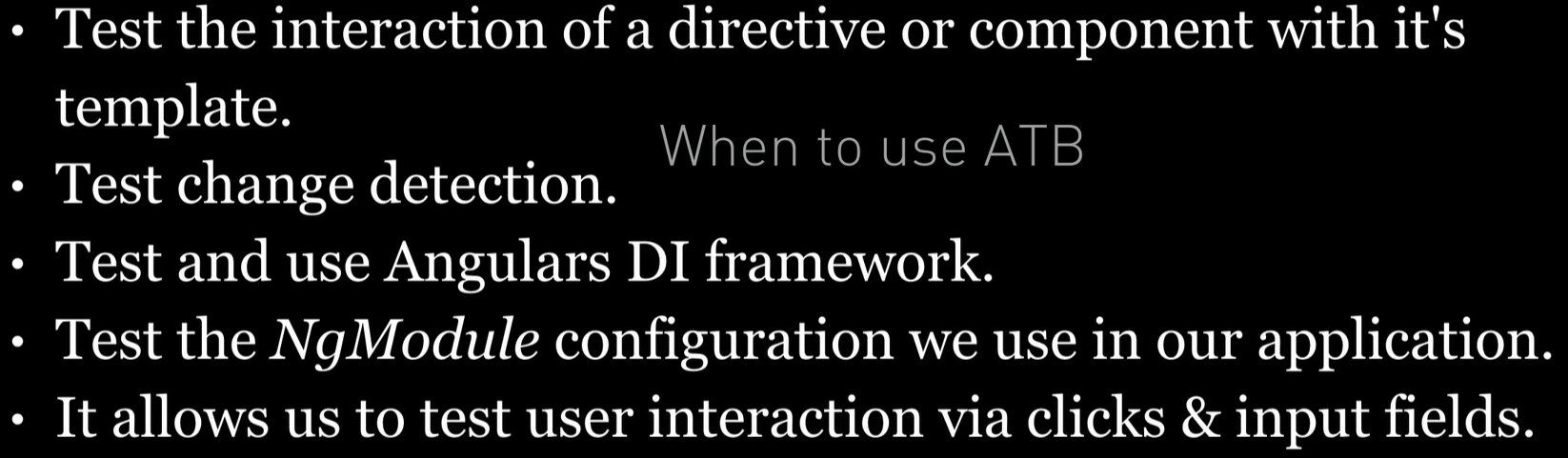
});

 **When to use ATB**

We will continue to use ATB for the rest of this section because:

* It allows us to test the interaction of a directive or component with it’s template.
* It allows us to easily test change detection and property binding in the component’s view.
* It allows us to test and use Angulars DI framework,
* It allows us to test using the NgModule configuration we use in our application.

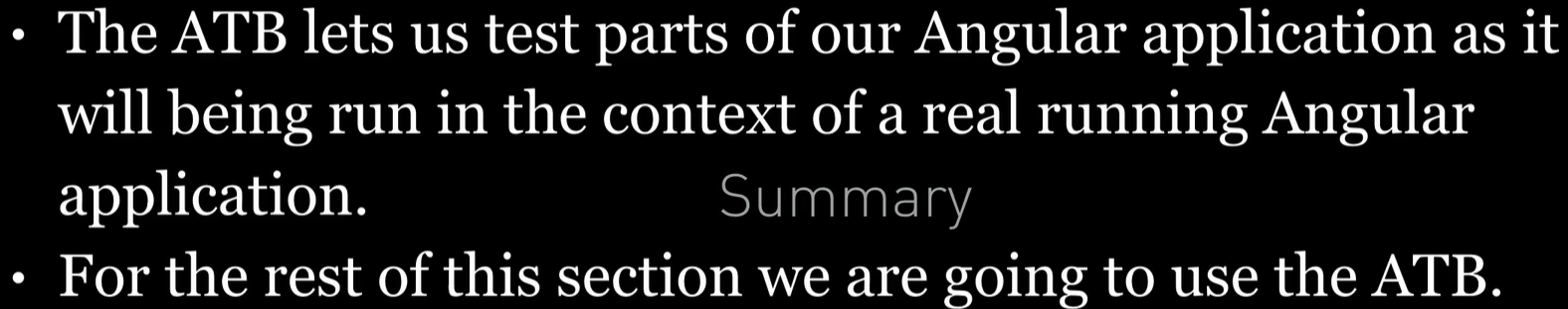
• It allows us to test user interaction via clicks & input fields



**Summary**

The ATB lets us test parts of our code as if it is being run in the context of a real Angular app.

It’s usefulness will become more apparent in future lectures, the next one being how to use the ATB to test change detection and property binding.



**Listing**

<http://plnkr.co/edit/jqw3ddMQU7zPQg9KBXJE?p=preview>