Add instructor notes here.



### What is Karma



- Karma Is A JavaScript test runner that fits the needs of an AngularJS developer
  - · Developed by the AngularJS team
  - · Can describe your tests with Jasmine, Mocha, Qunit
  - · Can write adapter for framework of your choice
- Karma allows you to test your code on real devices
- You can test your code on real browsers and real devices such as phones, tablets or on a headless PhantomJS Instance.
- Karma is a JavaScript test runner which means it will read code in a certain format (in our case written in <u>Jasmine</u>) and run it as a suite of tests

July 30, 2018 Proprietary and Confidential -2 -

- Unit Test
  - In unit tests component is tested in isolation without external recourses like file system and database and API
  - In angular terms testing angular component without template ,recourses, browser , API.
  - Ex angular component is using service to talk to API end point .then instead of service we can use Fake service instance.
  - Will focus functionality of component. Or Fake router.
- Integration Test
  - -We test the component with external resources like file system ,database, and API end point
- · End To End Test
  - -we test application in the browser -very slow and fragile.

# Importance of combining Karma with Angular



- Manually running Jasmine tests by refreshing a browser tab repeatedly in different browsers every-time we edit some code can become tiresome.
- Karma is a tool which lets us spawn browsers and run jasmine tests inside of them all from the command line.
- The results of the tests are also displayed on the command line.
- Karma can also watch your development files for changes and re-run the tests automatically.

July 30, 2018 Proprietary and Confidential - 3 -

Note:- Naming convention to be followed for writing angular unit test cases.

-If component name is "welcome.component.ts" then test file name should be "welcome.component.spec.js

- We use angular-cli ng-test command to run the unit test cases.

# Importance of combining Karma with Angular



- Karma lets us run jasmine tests as part of a development tool chain which requires tests to be runnable and results inspect able via the command line.
- It's not necessary to know the internals of how Karma works. When using the Angular CLI it handles the configuration for us and for the rest of this section we are going to run the tests using only Jasmine.

July 30, 2018 Proprietary and Confidential - 4 -

### Installation Of Karma



- Install Angular-cli command using
- C:\Users\users\Desktop\KarmaDemo>npm install -g angular-cli
- Go To the KarmaDemo folder and download sample application from internet or copy the sample demo
- C:\Users\users\Desktop\KarmaDemo>
- After that run npm install command which will install none
- C:\Users\users\Desktop\KarmaDemo>npm install
- After that test the application using following command
- C:\Users\users\Desktop\KarmaDemo> npm test

Proprietary and Confidential -5 -

**.Set up Karma**This setup is done through the common interface of the Karma configuration file called karma.conf.js.

This file will tell Karma which locations the tests are in, which reporting mechanism to use, and which browser to use to execute the tests. Here is an example Karma config file from the Angular CLI project:

```
basePath: ",
frameworks: ['jasmine', 'angular-cli'],
plugins: |
 require('karma-jasmine'),
 require('karma-chrome-launcher'), require('karma-remap-istanbul'),
 require('angular-cli/plugins/karma')
files:
 { pattern: './src/test.ts', watched: false }
preprocessors: {
    './src/test.ts': ['angular-cli']
remapIstanbulReporter: {
 reports: {
   html: 'coverage',
   Icovonly: './coverage/coverage.lcov'
angularCli: {
 config: './angular-cli.json',
 environment: 'dev'
},
///...
```

# Testing with Karma- What is TestBed



- · It is a angular testing utility.
- · It is important testing utility
- · Creates an Angular testing module
- Used to test interaction between a component and its template and different components
- It is present in the Package -'@angular/core/testing'

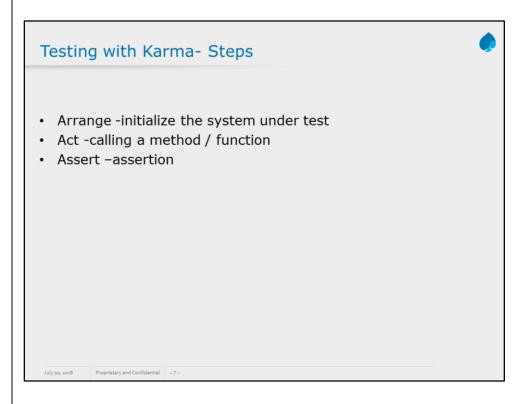
### TestBed.configureTestingModule(metadataObject)

- · Used to create a dynamic Angular testing module
- · Takes an @NgModule-like metadata object
- · metadataObject' can have most of the properties of NgModule.
- The TestBed.configureTestingModule() method takes a metadata object that can have most of the properties of an @NgModule.

July 30, 2018 Proprietary and Confidential -6 -

In Angular 2 in order to test any component we need to load it into a module beforehand.

This is done (in part) by a handy built-in helper class called <u>TestBed</u>.



# Testing with Karma-Steps



- beforeEach(function)
  - -Run some shared setup before each of the specs in the describe in which it is called
- afterEach(function)
  - -Run some shared teardown after each of the specs in the describe in which it is called
- · beforeAll(function)
  - -Run some shared setup once before all of the specs in the describe are run
- afterAll(function)

Run some shared teardown once after all of the specs in the describe are run

July 30, 2018 Proprietary and Confidential -8 -

# Testing with Karma-Test Bed and ComponentFixture Steps



- The general pattern to stand up a component test is as follows:
- Define a TestBed by calling the static method configureTestingModule.
- Start a test method and request the test bed by calling getTestBed() from the Testing module
- Ask the test bed to compile the component definitions with compileComponents()
- Inject the components by calling testBed.createComponent(ComponentClassName)
- Set properties and call methods on the component's class with the fixture.componentInstance property
- If needed, move the change detection forward with fixture.detectChanges()
- Ask the fixture for a nativeElement property and check the data on the DOM using a query selector

July 30, 2018 Proprietary and Confidential - 9 -

In Angular 2 in order to test any component we need to load it into a module beforehand.

This is done (in part) by a handy built-in helper class called <u>TestBed</u>.

# Testing with Karma- Test Bed and ComponentFixture Steps



- Remember, the ComponentFixture returned from TestBed.createComponent is your component under test, with several additional properties and methods:
- detectChanges() force a walk of the data graph and update the view with relevant content, and vice versa. This is a topdown one-time process, similar in result to the Angular 1 \$scope.\$digest() function but much faster
- componentInstance the actual component under test you can call methods of the component here
- nativeElement the DOM element of the template your component owns
- The TestBed was added around RC4 or RC5, and is the preferred method of testing components and services in the container.

July 30, 2018 Proprietary and Confidential -10 -

# Using Karma with Angular-TestBed API



### TestBed.createComponent(component)

- Used to create an instance of component-under-test
- Returns component test fixture

### ComponentFixture

- · Wrapper around a component
- Gives access to component instance as well as its template (DOM representation)
- ComponentFixture.componentInstance
  - Returns instance of the component class
- ComponentFixture.nativeElement
  - Returns the native DOM element at the root of the component
- · ComponentFixture.debugElement

Proprietary and Confidential - 11 -

- Provides a wrapper object around component's root native element
- Provides useful methods for querying the DOM

import { async, TestBed } from '@angular/core/testing';
import { SomeComponent } from './some.component';
beforeEach(() => {
 TestBed.configureTestingModule({ declarations: [ SomeComponent ],
 imports: [ // HttpModule, etc. ], providers: [ // { provide: ServiceA, useClass:
 TestServiceA } ] }); });
it('should do something', async(() => {
 // Overrides here, if you need them

TestBed.overrideComponent(SomeComponent, { set: { template: '<div>Overridden

TestBed.compileComponents().then(() => { const fixture = TestBed.createComponent(SomeComponent);

// Access the dependency injected component instance
const app = fixture.componentInstance; expect(app.something).toBe('something');

// Access the element const element = fixture.nativeElement;

template here</div>' // ... } });

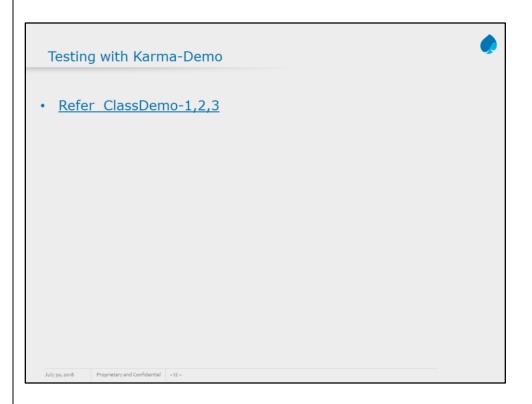
// Detect changes as necessary
fixture.detectChanges();
expect(element.textContent).toContain('something'); }); }));

A fixture is a wrapper for a component and it's template.

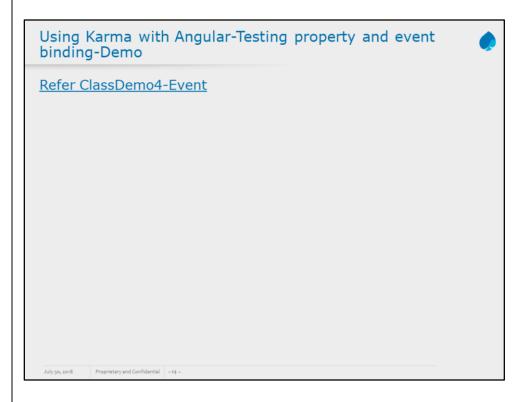
We create an instance of a component fixture through the TestBed, this injects the AuthService into the component constructor.

We can find the actual *component* from the componentInstance on the fixture.

We can get resolve dependencies using the TestBed injector by using the get function.



# DebugElement.query(predicate) Used to query the DOM predicate is a function that returns true if a condition is met Returns the first element that matches the predicate By.css() Predicate for use with DebugElement'squery functions Match elements by the given CSS selector ComponentFixture.detectChanges() Trigger a change detection cycle for the component DebugElement.triggerEventHandler('eventName', eventObj) Used to trigger an event on an element. For e.g., to invoke 'click' event on a button with id 'save', following code is used -constbutton = fixture.debugElement.query(By.css('#save')); -button.triggerEventHandler('click', null);



# Using Karma with Angular-Handle Component Dependency



# **Providing Dependencies**

- -Register the service in the testing module by adding it to the 'providers' array
- -Register any other Angular dependency in the testing module by adding it to 'imports' array

For e.g., if the service internally uses Http, add HttpModule to 'imports' array of testing module configuration

# **Getting Dependencies**

- -TestBed.get(service)
- -Returns a reference to 'service' instance injected in a component  $% \left( 1\right) =\left( 1\right) \left( 1\right) +\left( 1\right) \left( 1\right) \left( 1\right) +\left( 1\right) \left( 1\right) \left( 1\right) \left( 1\right) +\left( 1\right) \left( 1$

July 30, 2018 Proprietary and Confidential -15 -

# Using Karma with Angular-Handle Component Dependency



Identify the dependencies and their methods that are used within the component

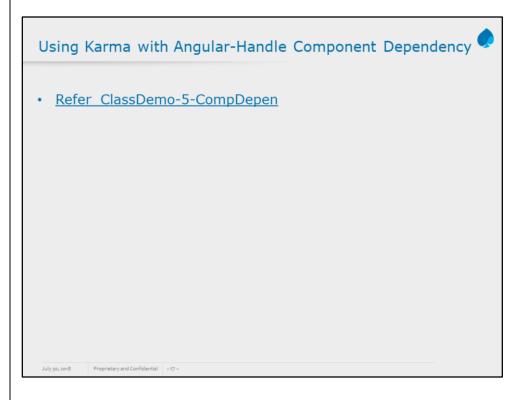
- -Create a stub class for each of the dependencies
- -Define method stubs
- -Replace the actual dependency with their corresponding stub implementation within the 'providers' array

TestBed.configureTestingModule({

})

]

Proprietary and Confidential -1



# Using Karma with Angular-Handle Routing Components



- We use RouterTestingModule to provide our test route.
- This uses the spy implementation of Location which does not trigger request for new url but does let us know the target URL So that we can use it in our test spec.

July 30, 2018 Proprietary and Confidential -18 -

