



Jaymin Padaira
Full Stack Developer

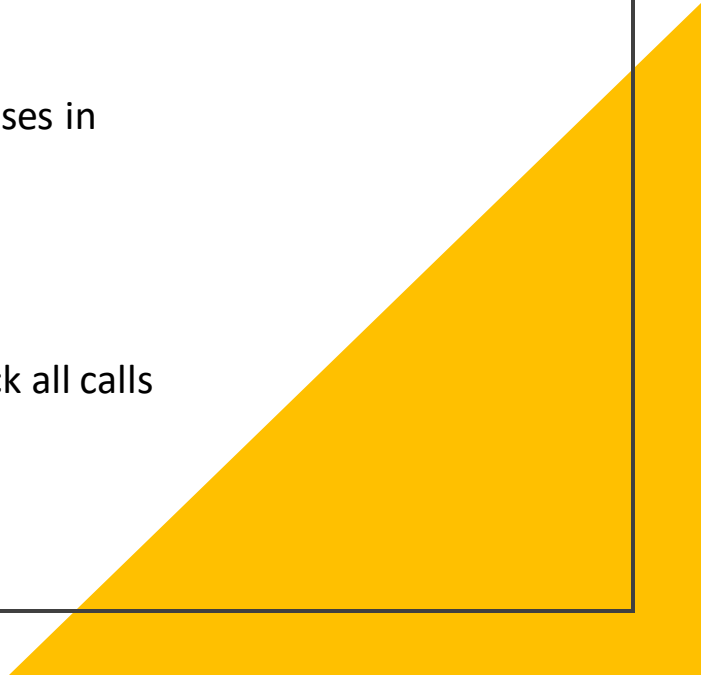
Reference: <https://www.udemy.com/course/angular-unit-test-case-with-jasmine-karma>

Agenda

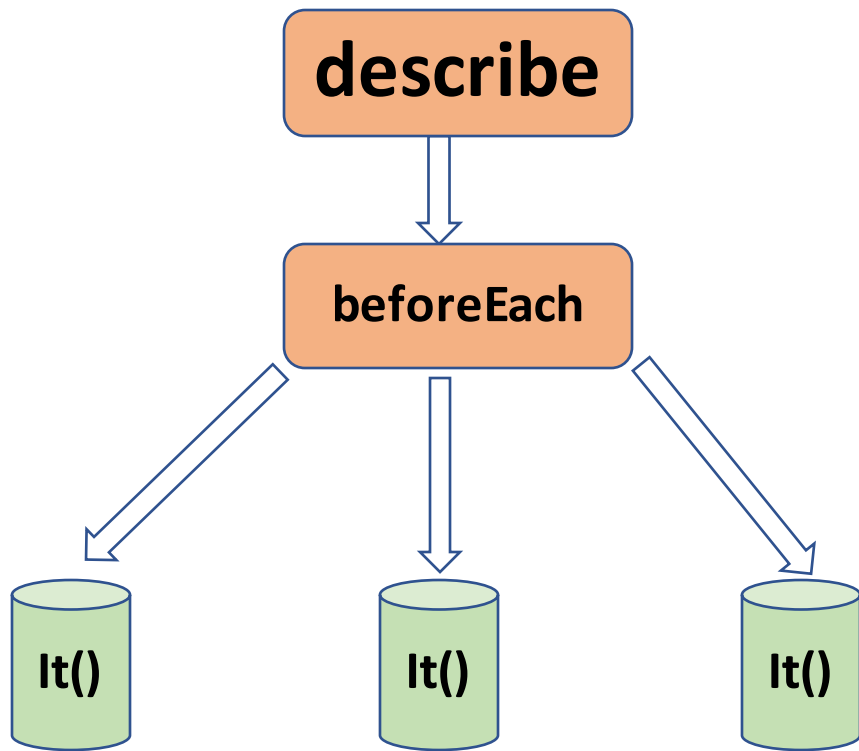
- What is Jasmine & karma?
- Flow of unit test case.
- Jasmine & karma configuration with angular.
- How to run unit test cases.
- Write unit test cases.

What is Jasmine?

- Jasmine is an open-source JavaScript testing framework.
- Framework provides building blocks to write JavaScript unit test.
- It means you can cover each line of code to write unit test.
- It can be used to test any type of JavaScript application.
- Jasmine is a behavior-driven development (BDD) framework.
- BDD is a software development approach that allows the developer to create test cases in simple text (non-technical) language.
- It does not depend on any other JavaScript framework.
- Jasmine supports asynchronous testing.
- Jasmine has the test double functions call spies, a spy can stub any function and track all calls to it and its arguments.



Flow of unit test case



- The Angular testing package includes two utilities called TestBed and async.
- TestBed is the main Angular utility package.
- The describer container contains different blocks (It, beforeEach, xit, afterEach, etc.)
- beforeEach runs before any other block. Other blocks do not depend on each other run.

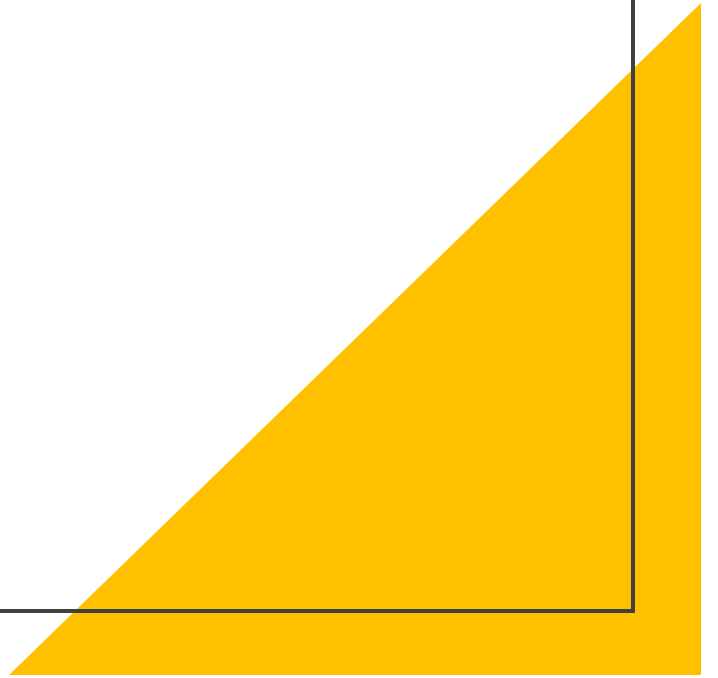
Jasmine & karma configuration with angular

- It create auto when we create project.
- It add in package devDependency
 - jasmine-core // allow to write test case
 - Karma //mission runner
 - karma-chrome-launcher // run testcase in chrome
 - karma-coverage // check code coverage
 - karma-jasmine
 - karma-jasmine-html-reporter
- In **angular.json** file has block **test** in that it set config file path
 - Main : "src/test.ts" //this is entry point of test when we run test it start with this file
 - KarmaConfig: "karama.conf.js"

How to run unit test cases

- In terminal run "**ng test**" command used.
- If you want to run particular folder use "**ng test --include src/app**" command used.
- This run all "***.spec.ts**" files.
- We need write all testcases in spec file.
- Default when we create any service, pipe, module or component it create with spec file also so we can write testcase inside that file.

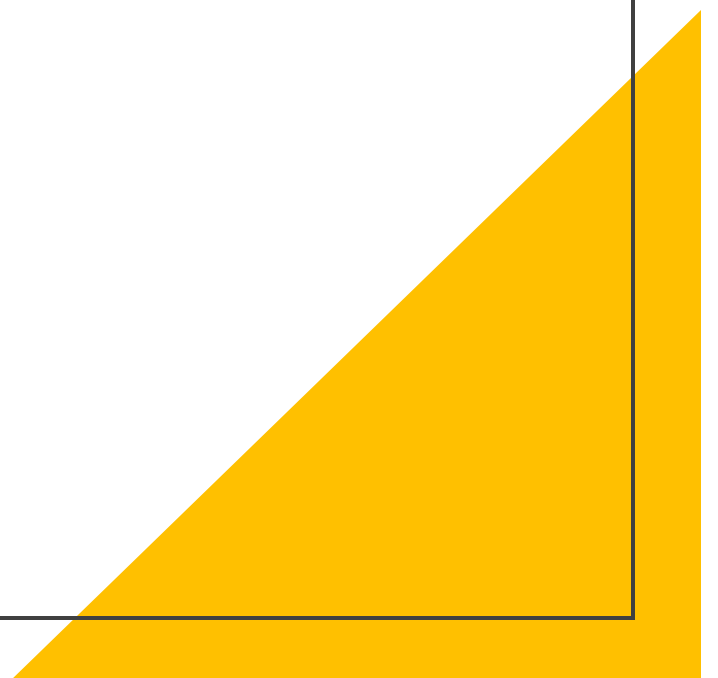
Start to write unit test cases



Exclude Angular unit test case from execution

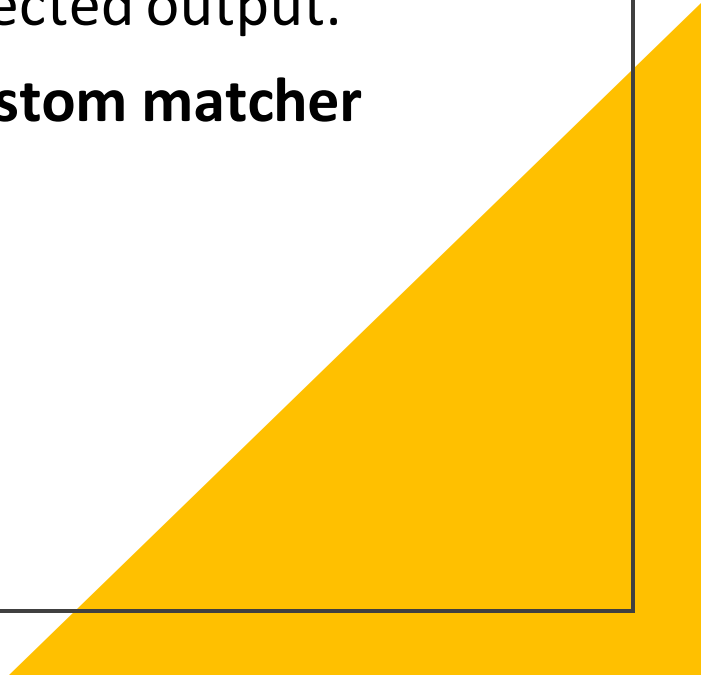
- Case using 'x' keyword we can exclude any test case from execution.
- Example
 - `xdescribe('describe test exclude execution', () => {})`
 - `xit('it test exclude execution', () => {});`

Jasmine Matchers



What are matchers

- Matchers are nothing but compare functions to compare the expected and actual result in the test specs.
- Matchers are the JavaScript function that dose a Boolean comparison between an actual output and expected output.
- There are two type of matchers **Inbuild matcher** and **Custom matcher**



Inbuild Matchers

Ref :: [Namespace: matchers \(jasmine.github.io\)](https://jasmine.github.io/3.5.0/matchers.html)

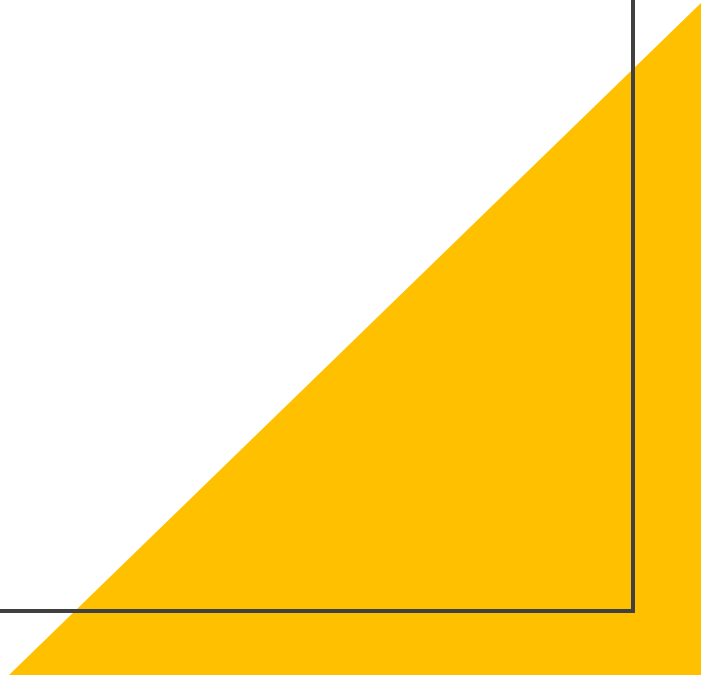
A large yellow triangle is positioned in the bottom right corner of the slide, pointing towards the top right.

- `toBe` // number, string, boolean (`===`)
- `toEqual` // object
- `toBeTruthy` // return true check
- `toBeFalsy` // return false check
- `toBeTruthy` // any data came as success
- `toBeFalsy` // return as false data
- `toBeGreaterThan` // `>`
- `toBeGreaterThanOrEqual` // `>=`
- `toBeLessThan` // `<`
- `toBeLessThanOrEqual` // `<=`
- `toMatch` // regex
- `toBeCloseTo` // close to that number with precision or not

- toBeDefined // it's defined or not anything like function, variable, object, etc...
- toBeNull // it check value is null
- toContain //check value exist in array just like filter array get data
- toBeNaN // check return value is null
- ToBeInstanceOf // check data type
- toHaveBeenCalled
- toHaveBeenCalledBefore
- toHaveBeenCalledOnceWith
- toHaveBeenCalledTimes
- toHaveBeenCalledWith
- toHaveClass
- toHaveSize
- toThrow
- toThrowError
- toThrowMatching

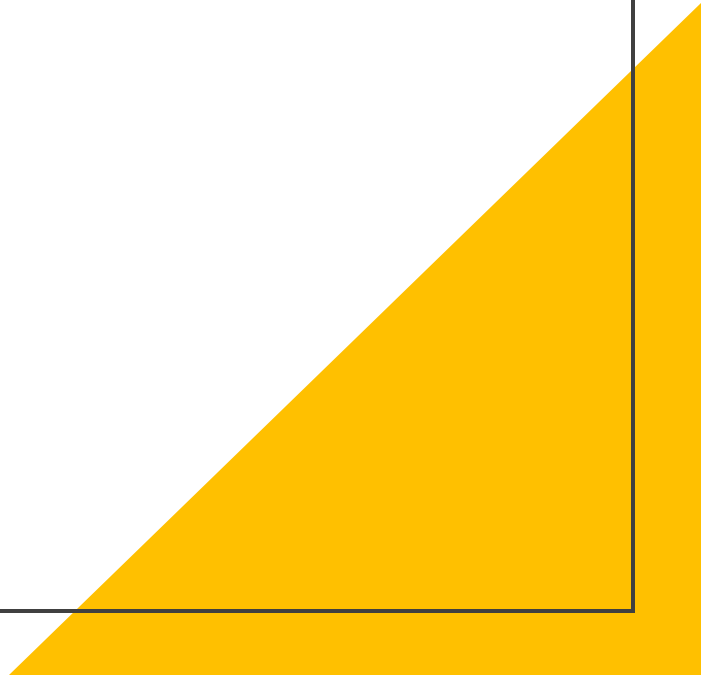
Custom matchers

- The matchers which are not present in the inbuilt system library of jasmine is called custom matcher.
- Custom matcher need to be defined explicitly().



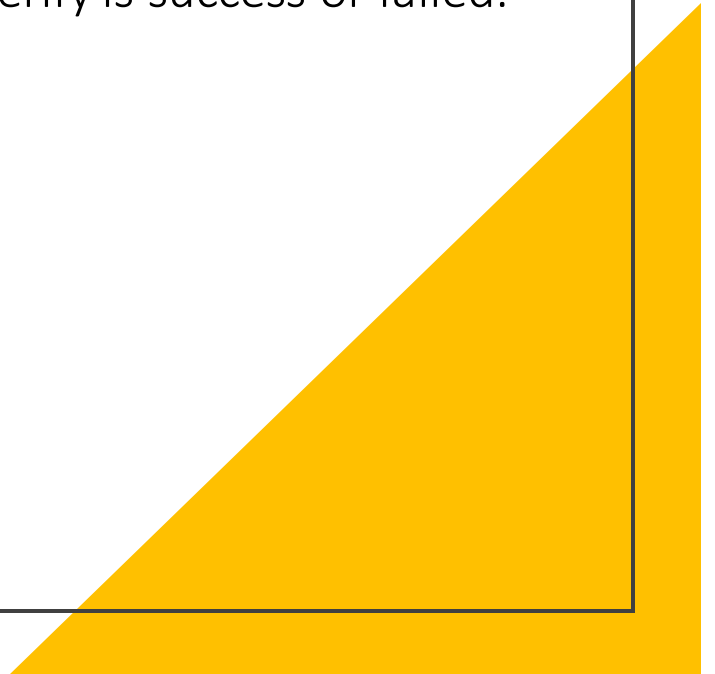
Jasmine Functions

- **beforeEach()** is run before each test in a describe
- **afterEach()** is run after each test in a describe
- **beforeAll()** executes once and only once for the describe block containing it, just before the any block run.
- **afterAll()** executes once and only once for the describe block containing it, just after the any block run.



What is Arrange-Act-Assert (AAA) Pattern


- Arrange-Act-Assert is great way to structure test cases.
 - **Arrange** : inputs and targets. Arrange steps should be setup the test case.
 - **Act** : on the target behavior. Act steps should cover the main thing to be tested.
 - **Assert** : expected outcomes. Act steps should return response verify is success or failed.



TestBed and Component Fixture

- The Angular Test Bed (ATB) is higher level Angular Only testing framework that allows us to easily test behaviors that depend on the Angular Framework.
- We still write our tests Jasmine and run using Karma, but we now have a slightly easier way to create component, handle injection, test asynchronous behavior and interact with our application.

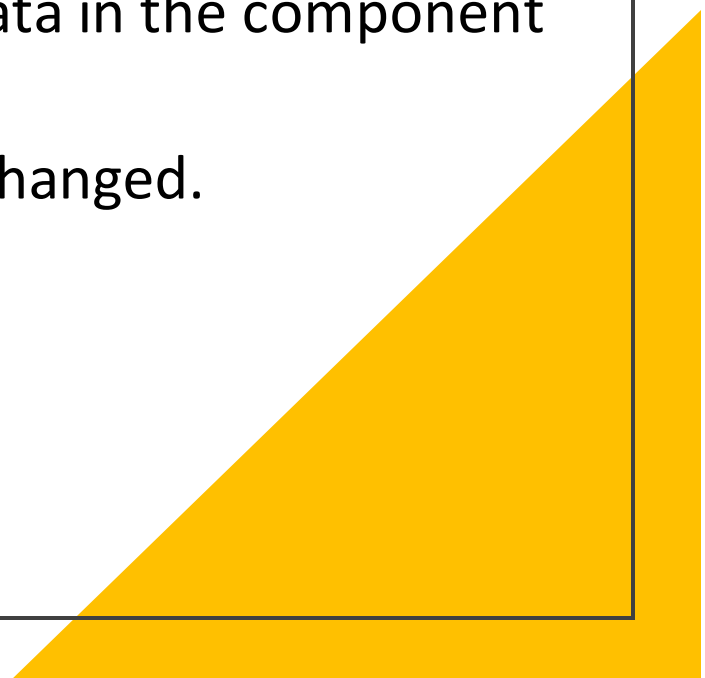
When to use ATB?

- It allows us to test the interaction of directive or component with its template.
 - It allows us to easily test change detection.
 - It provide methods to create components and services for unit test case.
 - It allows us to test and use Angular's 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 and input fields.
- 
- A large yellow triangle is positioned in the bottom right corner of the slide, pointing towards the top right.


SpyOn to mock and Stub Methods

- Jasmine Spice help us to mock the execution of the angular method.
- It's easy way to check a method was called or not, without leaving Subject Under Test (SUT).
- We can chain the SpyOn method to get dummy return value using `".and.returnValue()"`.
- SpyOn can call the original function using `".and.callThrough()"`.
- Methods::
 - Stub: a dummy piece of code, that lets the test run, but you don't care what happens to it.
 - Mock: a dummy piece of code, that you VERIFY is called correctly as part of the test.
 - Spy: a dummy piece of code, that intercepts some calls to a real piece of code, allowing you to verify calls without replacing the entire original object.

Change Detection

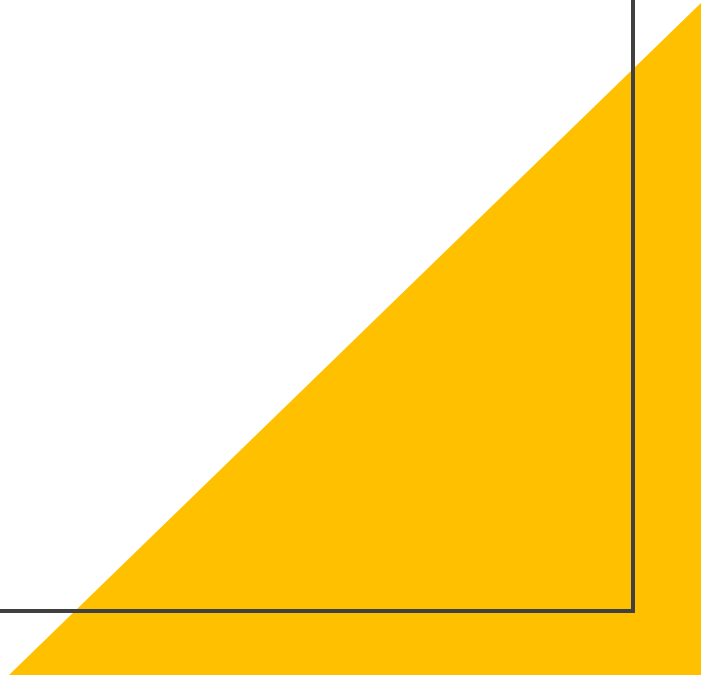
- Change detection is the background of the Angular framework, each component has its own change detector.
 - Angular can detect when data changes in the component and can re-render the view to display the updated data. Angular makes sure that data in the component and the view are always in sync with each other.
 - Change detection means updating the view whenever data changed.
- 
- A large yellow right-angled triangle is positioned in the bottom right corner of the slide, pointing towards the top-left.

Debug Element & DOM events

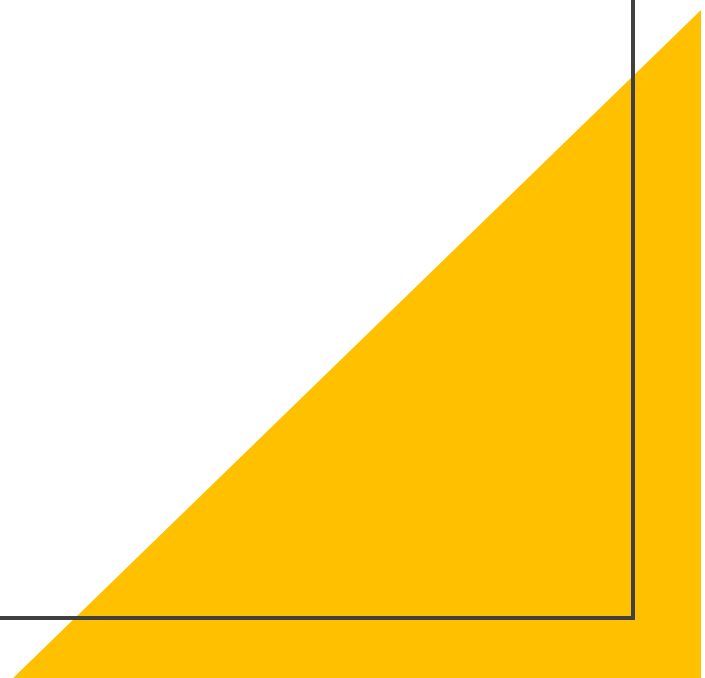
- Debug Element is an Angular class that contains all kinds of references and methods relevant to investigate an element as well as component.
 - Instead of creating an HTML element tree, Angular creates a Debug Element tree that wraps the native elements for the runtime platform.
 - The native element property unwraps the Debug Element and returns the platform-specific element object.
 - Native element returns a reference to the DOM element.
 - Trigger event handler is function that exists on Angular's Debug Element.
- 
- A large yellow triangle is positioned in the bottom right corner of the slide, pointing towards the top right.

Private method and private variable

- We use like `component["calculate"]()`, `component["Name"]`
- `spyOn<any>(component,"ShowName")` // use any for private method call for spyon

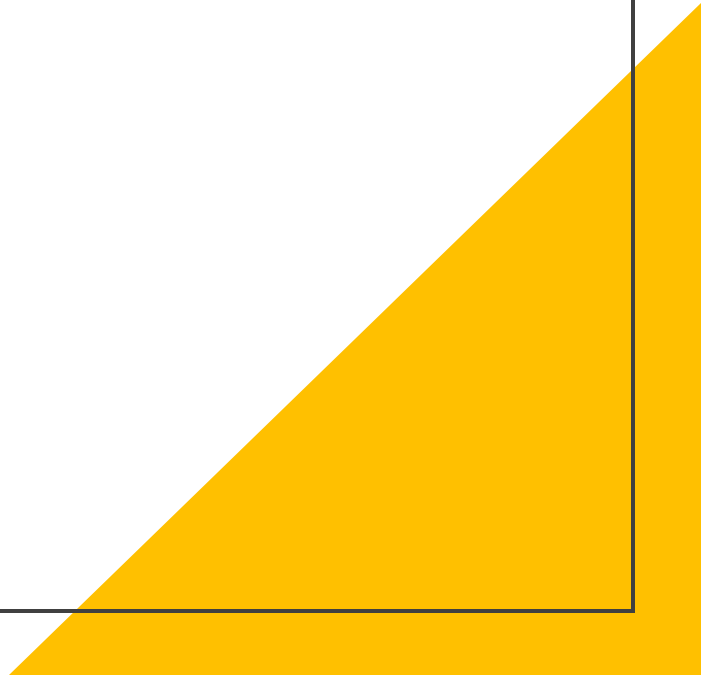


Interpolation

- Interpolation is a technique that allows the user to bind a value to UI element.
 - Interpolation binds the data on-way. This means that when value of the field bound using interpolation changes, it is updated in the page as well. It cannot change the value of the field.
 - Return type of interpolation is always a string.
- 
- A large yellow triangle is positioned in the bottom right corner of the slide, pointing towards the top right.

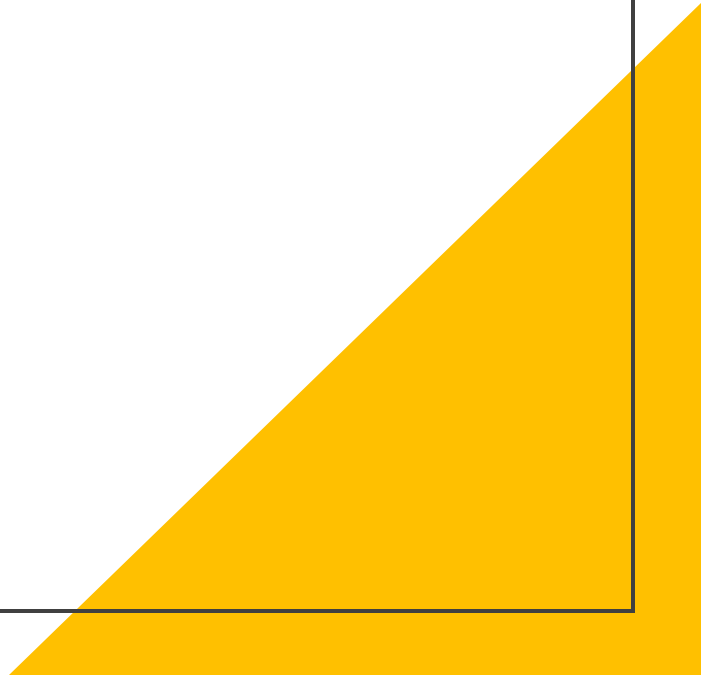
ngClass – ngStyle

- It's use
 - `element.getAttribute('class')`
 - `element.getAttribute('style')`



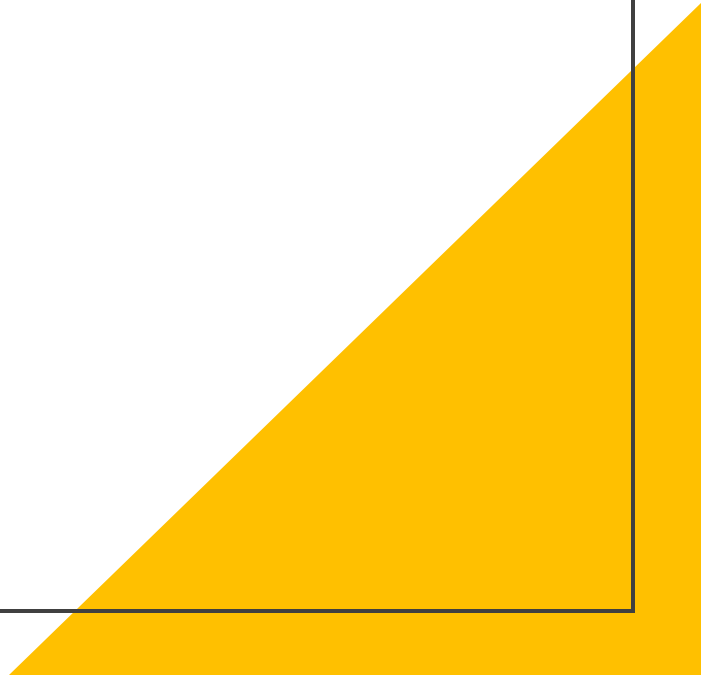
attribute-binding

- It's use
 - `element.getAttribute('colspan')`
 - `element.getAttribute('aria-label')`



event-binding

- It's use
 - `element.click();`
 - `element.dispatchEvent(new Event('input'));`



WhenStable, Async Test and twoWay data binding

- WhenStable

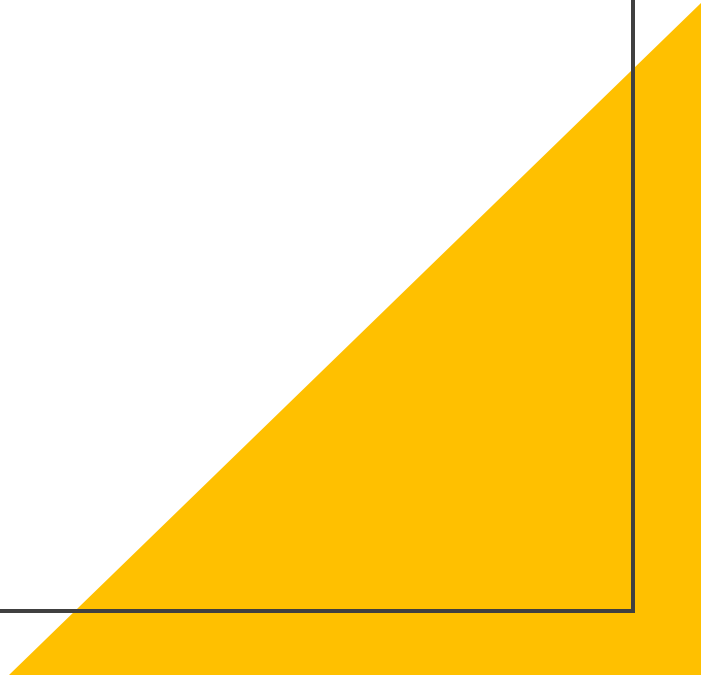
- The `fixture.whenStable()` return a promise that resolves when the JavaScript engine's task queue becomes empty. In this example, the task queue becomes empty when the observable emits the first quote.
- The `async` utility tells Angular to run the code in a dedicated test zone that intercepts promises. The use of `async` with `whenStable` allows us to wait until all promises have been resolved to run our expectations.
- The test resumes within the promise callback, which calls `detectChanges()` to update the quote element with the expected text.

fakeAsync and async

- The Jasmine **done** function and spy callbacks. We attach specific callbacks to spies so we know when promises are resolved, we add our test code to those callbacks and then we call the done function. This works but means we need to know about all the promises in our application and be able to hook into them.
- We can use the Angular async and whenStable functions, we don't need to track the promises ourselves, but we still need to layout code out via callback functions which can be hard to read.
- We can use the Angular fakeAsync and tick functions, this additionally lets us lay out our async test code as if it were synchronous.
- The problem with async is that we still have to introduce real waiting in our tests, and this can make our tests very slow. FakeAsync comes to the rescue and helps to test asynchronous code in synchronous way.

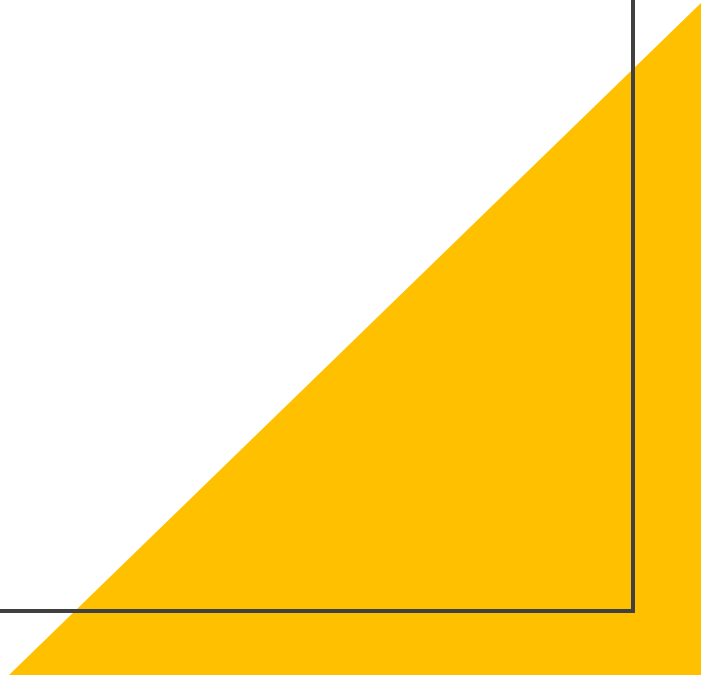
Conditional rendering

- NgIf
- ngSwitch



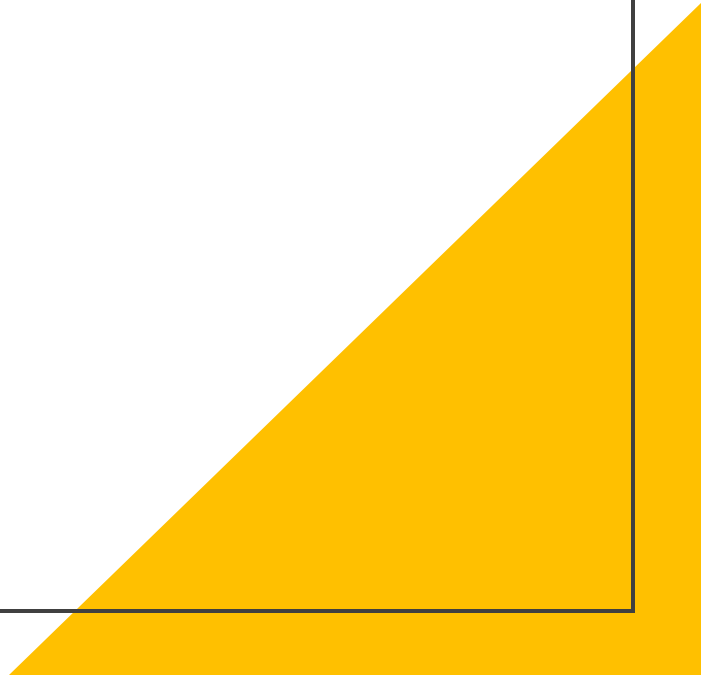
ngFor

- df



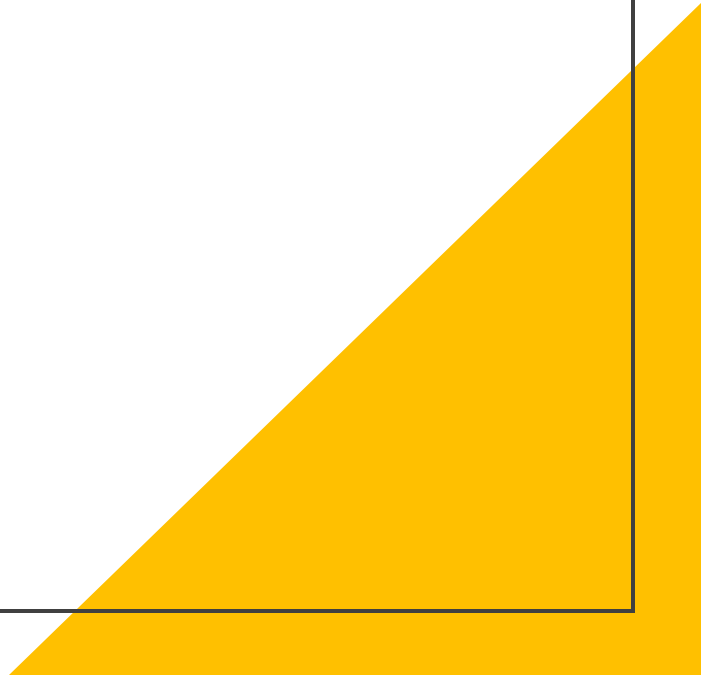
dependency-injection

- df

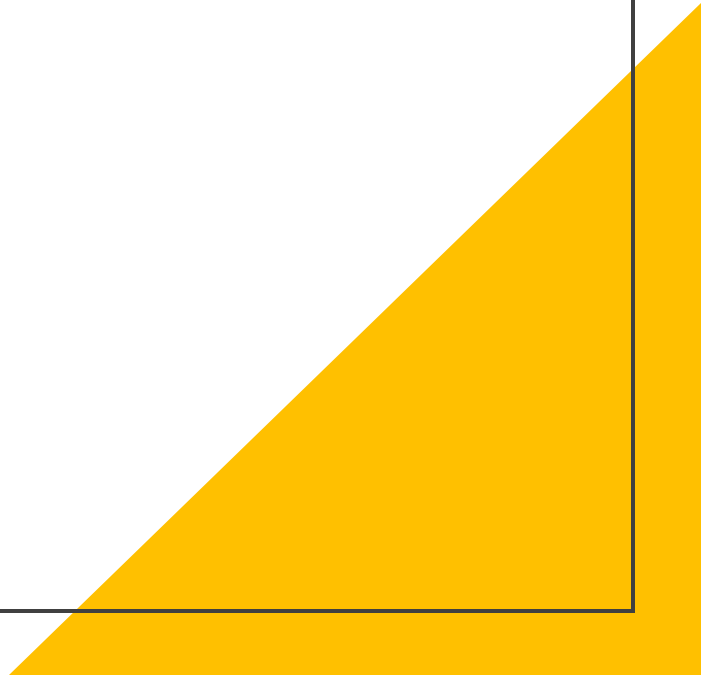


Pipe and custom pipe

- Pipes are referred as filters.
- It helps to transform data and manage data within interpolation, denoted by `{{|}}`.
- It accepts data, arrays, integers and strings as inputs which are separated by `'|'` symbol.

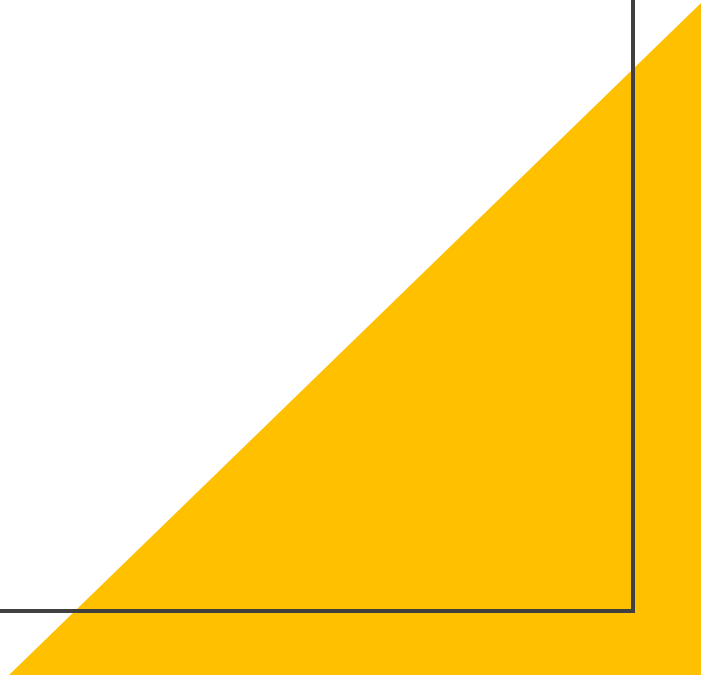


@Input and @Output



Routing in Angular

- Routing is an important key feature for every frontend framework.
- It is the process of dividing the UI of an application using URLs.
- It allows the developers to build modern single-page applications that can be loaded once by a browser and provide multiple views.



Lazy loading

- Lazy loading is technology of angular that allows you to load JavaScript components when a specific route is activated. It improves application load time speed by splitting the application into many bundles. When the user navigates by the app, bundles are loaded as needed.
- Lazy loading helps to keep the bundle size small, which helps reduce load times. We must use the class decorator to create an Angular
- Module `@NgModule`, and the decorator use a metadata defines the module.
- The main properties are:
 - Import: Components of this module are used with Array with other modules.
 - Declarations: It receives an array of the components
 - Export: Defines an array of components, directives and pipes used by other modules.
 - Provider: Declares services that are available to the entire application if it is a root module.

Http Client or service

- HttpClientTestingModule is used to test the http methods using httpClient.
- HttpClientTestingModule inject HttpTestingController.
- The HttpClientTestingModule allows you to easymock HTTP request by providing you with the Http TestingController service.
- Using the HttpClientTestingModule and Http TestingController provided by angular makes mocking out results and testing http requests simple by providing many useful methods for checking http requests and providing mock for checking http requests and providing mock responses for each request.

Http TestingController methods

- Abstract class Http TestingController {
- Abstract math(match: string | RRequestMatch | ((req: HttpRequest<any>)=> boolean)):TestRequest[]
- Abstract expextOne(url: string,description ?:string):TestRequest
- Abstract expextNone(url: string,description ?:string): void
- Abstract verify(opts?:{ ignoreCancelled?:boolean}): void
- }
- Match:- Search for requests that match the given parameter, without any expectations.

- ExpectOne:- Expect that a single request has been made which matches the given URL and returns its mock.
- ExpectNone:- Expect that no requests have been made which match the given URL.
- Verfiy:- Verify that no unmatched requests are outstanding.
- **Test Request**
- A mock requests that was received and is ready to be answered.

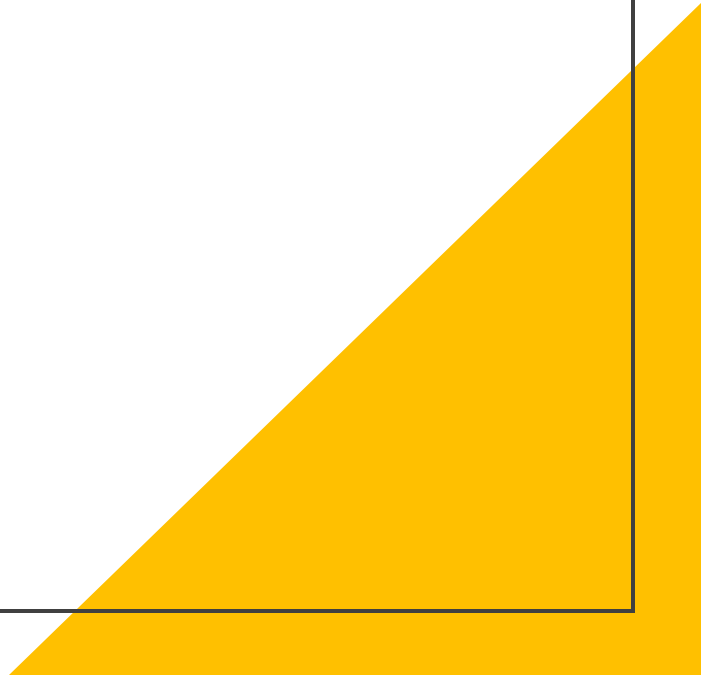
Property	Description
Cancelled: Boolean	Read-Only Whether the request was cancelled after it was sent.
Request: HttpRequest<any>	Declared in Constructor

- **Methods:-**

- **Flush()** :- Resolve the request by returning a body plus additional Http information (such as response headers) if provided. If the request specifies an expected body type, the body is converted into the requested type. Otherwise, the body is converted to Json by default.
- **Error()**:- Resolve the request by returning an errorEvent. Resolve the request by returning an errorEvent.
- **Event**:- Deliver an arbitrary HttpEvent on the response stream for this request.
- **Jsonplaceholder** – free fake rest Api
- **URL**:: for Fack Api call test
- <https://jsonplaceholder.typicode.com>.

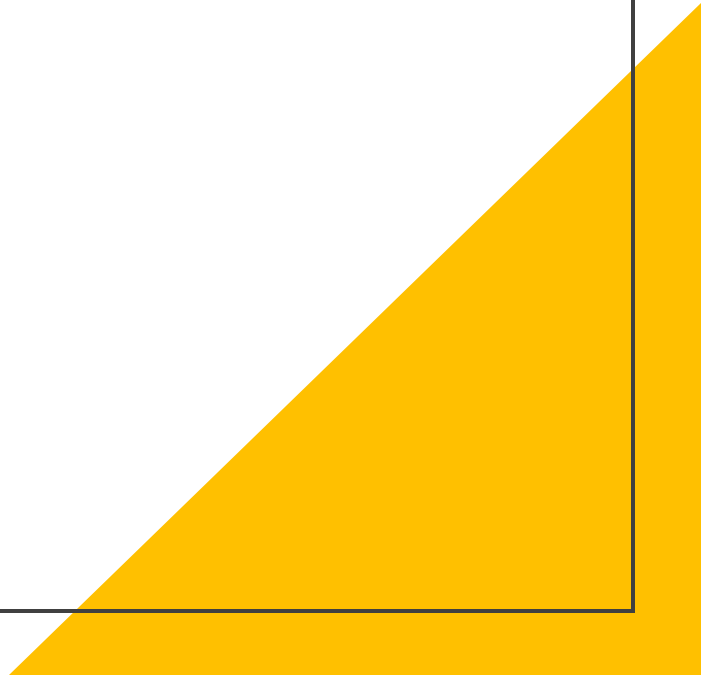
Subscribe method

- Unit test on call service method to call API in the controller.

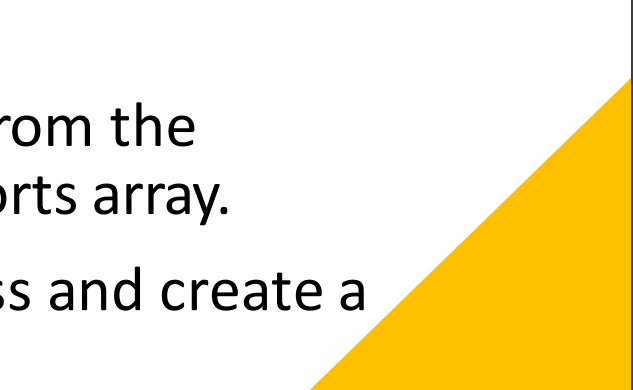


CreateSpy , createSpyObj and spyOn

- Jasmine.createSpy can be used when there is no function to spy on. It will track calls and arguments like a spyOn but there is no implementation.
- Jasmine.createSpyObj is used to create a mock that will spy on one or more methods. It returns an object that has a property for each string that is a spy.



Reactive Forms

- Reactive forms are forms where we defined the structure of the form in the component class.
 - We create the form model with Form Controls, and Form Arrays.
 - We also defined the validation rules in the validation rules in the component's class. Then, we bind it to the HTML form in the template
 - To use reactive form controls, import `ReactiveFormsModule` from the `@angular/form` package and add it to your `NgModule`'s imports array.
 - To register a single form control, import the `FormControl` class and create a new instance of `FormControl` to save as a class property.
- 
- A large yellow triangle is positioned in the bottom right corner of the slide, pointing towards the top right.

Template Driven Forms

- What is template drive form?
- Template-driven forms use two-way data binding to update the data model in the component as changes are made in the template and vice versa.
- Template driven forms are forms where we write logic, validation, controls etc. In the template part of the code. The template is responsible for setting up the form< the validation, control, group, etc.
- What are the differences between reactive forms and template driven forms?
- Template Driven Forms are based only on template directives, While Reactive forms are defined programmatically at the component class level. Reactive forms are a better default choice for new applications, as they are more powerful and easier to use.

Code Coverage

- Code coverage, also called test coverage, tell you which parts of your code are executed by running the unit and integration test code coverage is typically expressed as percent values, for example,
 - 79% Statements
 - 53% branches
 - 74% functions
 - 78% lines
- To generate a coverage report run the following command in the root of your project.
 - `ng test --code-coverage`
 - `ng test --include src/app --code-coverage`
- If you want to create code-coverage reports every time you run the test, set the options in "angular.json"
 - `"test":{"options":{"codeCoverage":true}}`