Angular Testing

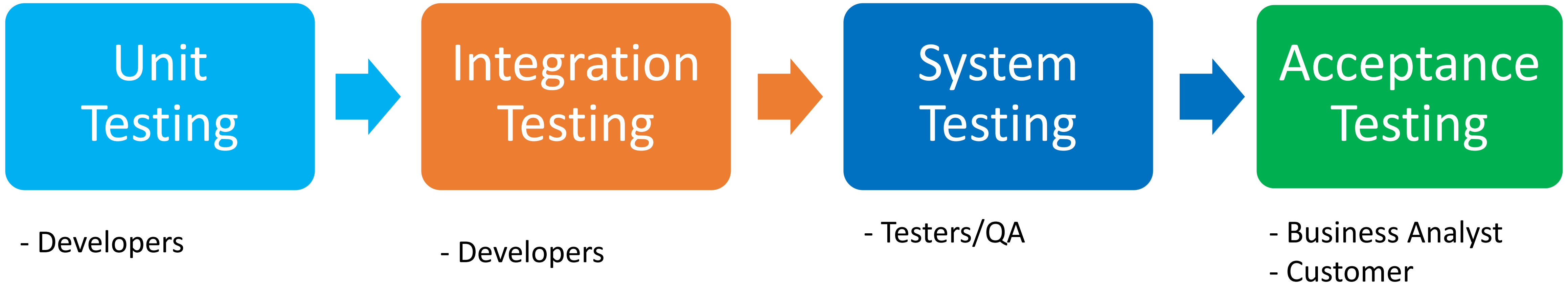
Agenda

- What is Testing?
- Types of Testing?
- What is Unit Testing?
- Angular CLI Setup and Angular Testing Tools
- Jasmine Test Spec
- Setup and Teardown
- Testing A Component
- Debugging and Code Coverage

What is Testing?

- Testing is a process to validate and verify a system or it’s components to determine whether it meet the specified requirements or not.
- Helps you to identify bugs, any gap or missing requirement, so that you can fix them and develop a quality product.

Types of Testing



What is Unit Testing?

- A unit testing is a method, where each unit or component of a software is tested to determine whether it is fit for use or not
- A single unit is any block of code (i.e. function or class) that has one and only one responsibility
- A function might have multiple unit tests according to the uses and output of the function

Angular CLI setup

Node 6.9+

www.nodejs.org

Angular CLI

```
npm install -g @angular/cli
```

Create New Project

```
ng new appName
```

Run Test

```
ng test
```

Angular Testing Tools

Test Runner

Karma

Test Framework

Jasmine

Test Utilities

Angular (TestBed, ComponentFixture)

Angular Testing Tools (Contd.)

- Karma – A test runner for running unit tests.
- Jasmine – A framework for writing basic tests. It ships with an HTML test runner that executes tests in the browser.
- Test Utilities – Angular provides TestBed class and several helper functions for unit tests.
  - TestBed is used to configure & initialize environment for unit testing.
  - Provides methods for creating components and injecting services in unit tests

Jasmine Test Spec

describe(str, fn)	it(str, fn)	expect(actual)	matcher(expected)
• A Test Suite • Contains Test Specs	• A Test Spec • Contains 1 or more test expectations	• An expected piece of behavior	• Does a boolean comparison • toEqual, toContain, toBeNull

Jasmine Test Spec Contd.

- The *describe(string, function)* defines a collection of *Test Specs*, called as Test Suite
- The *it(string, function)* defines a *Test Spec*, containing one or more Test expectations
- The *expect(actual)* describes an expected piece of behaviour in the application
- The *matcher(expected)* like toEqual, toContain, toBeNull does a boolean comparison of the *expected* value with *actual* value
- You can do negative assertion with not

```
describe('Hello world', () => {
    it('says hello', () => {
        expect(helloWorld())
        .toEqual('Hello world!');
    });
    it('not  equal to Hello', () =>
    {
        expect(helloWorld())
        .not.toEqual('Hello!');
    });
});
```

Setup and Teardown(cleaning up)

beforeAll()

- Called once, before all the specs in a test suite run

beforeEach()

- Called before each test spec run

afterAll()

- Called once, after all the specs in a test suite finished

afterEach()

- Called after each test spec run

DotNetTricks

Setup and Teardown(cleaning up) Contd.

- beforeAll - This function is called once, before all the specs in a test suite has been run
- beforeEach - This function is called before each test spec has been run
- afterAll - This function is called once, after all the specs in a test suite has been finished
- afterEach - This function is called after each test spec has been run

```
describe('Hello world', () => {
  let expected = "";
  beforeEach(() => {
      expected = "Hello World";
            });
  afterEach(() => {
            expected = "";
            });
  it('says hello', () => {
            expect(helloWorld())
            .toEqual(expected);
            });
});
```

Getting Started

DotNetTricks

Testing A Component

DotNetTricks

Debugging

Code Coverage

Testing A Component with Template

DotNetTricks

Testing A Component with Dependencies

Testing Http Service

DotNetTricks

Learn. Build. Empower

DotNetTricks

Thank You!

DotNetTricks