# Angular Fundamentals



# What is Angular

- A JavaScript framework
- For building client-side applications
- Using HTML, CSS and TypeScript

# Why Angular?

- Expressive HTML
- Powerful Data binding
- Modular by Design
- Built-in Back-end integration

## Structure of Angular Application

Application = Component + Component + ••••

Services

## What is a Component?

Component

**Template** 

HTML Layout,
Includes binding
& directives

Class

Properties

Methods

Metadata

- Typescript code supporting View
- Properties representing data
- Methods representing logic
- Extra data for angular
- Defined with decorators

## Prerequisites

#### Required

- JavaScript basics
- HTML basics
- CSS

#### Helpful

- OOP Concepts
- ✓ C++, C#, JAVA

### > Not Required

- Angular Knowledge
- ✓ Typescript Knowledge

# Typescript

- TypeScript is the programming language we use when building Angular applications
  - Open-source language
  - Superset of JavaScript
  - Transpiles to plain JavaScript
  - Strongly typed
  - Class-based object-orientation

## What component contains?

Component

Е

**Template** 

+

Class

**Properties** 

Methods

+

Metadata

- ✓ View layout
- ✓ Created with HTML
- ✓ Includes binding and directives

- Code supporting the view
- ✓ Created with TypeScript
- ✓ Properties: data
- ✓ Methods: logic

- ✓ Extra data for Angular
- ✓ Defined with a decorator

## Service

- Class with focused purpose
  - ► Independent from any component
  - ▶ Provide shared data or logic across component
  - ► Encapsulate external interactions

## **How Service Works**

#### Service

export class myService {}

### Component

let svc = new myService();

svc

## How Service Works cont...



Service
export class myService {}

Component

constructor(private myService) {}

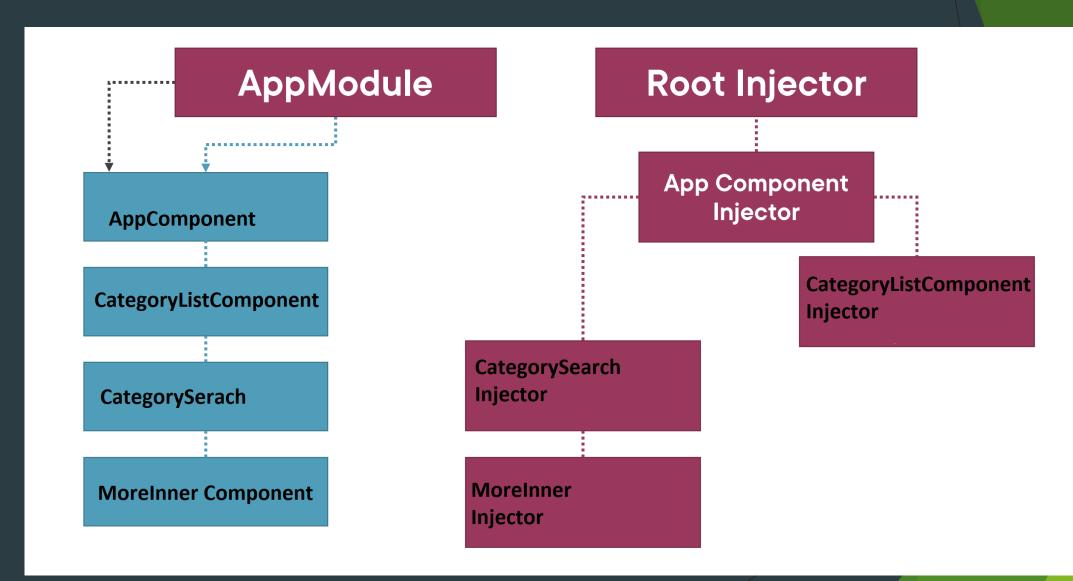
## Dependency Injection

A coding pattern in which a class receives the instances of objects it needs (called dependencies) from an external source rather than creating them itself

# **Building Service**

- Create Service Class
- Define metadata with decorator
- Import what we need

## Angular Injectors



## Registering Service

#### **Root Injector**

Service is available throughout the application

Recommended for most scenarios

### **Component Injector**

Service is available ONLY to that component and its child (nested) components

Isolates a service used by only one component

Provides multiple instances of the service

## Injecting Service

```
product.service.ts
@Injectable({
  providedIn: 'root'
export class ProductService { }
 product-list.component.ts
@Component({
  templateUrl: './product-list.component.html',
  providers: [ProductService]
export class ProductListComponent { }
      app.module.ts
@NgModule({
  imports: [ BrowserModule ],
  declarations: [ AppComponent ],
  bootstrap: [ AppComponent ],
  providers: [ProductService]
export class AppModule { }
```