# Angular

Angular is a framework for building modern single-page applications.( client side applications )

Traditional application each user action results in a full HTML page load. So if you do any work in the browser it'll make a request at a server and give you a full page every time.

However in a single-page application,a web application is composed of a single page and based on the user actions the application page is updated so it normally performs a partial update instead of a full page load.

i.e parts of the view get refreshed asychronously without having to reload the entire page.

Why Angular ?

* Modular approach
* Reusable
* Maintainable

# AngularJS

The first version of AngularJS 1.0 was released in October 2010 .

Angular team found ways that they could actually improve the framework and actually make it better and so in this case they actually did a total rewrite of the framework so they released Angular 2 in October 2016.

It's a total rewrite of the framework. It's a clean break from AngularJS 1.0. So it's actually two separate frameworks, two separate code bases and so it was a big transition there between the two.

So using Angular 2 in going forward then all of those newer versions they simply followed the same framework approach and they made incremental improvements so they released Angular 4, 5, 6, 7 and 8.

No major changes here just incremental improvements. But here the key is that AngularJS 1.0 and Angular 2 a total rewrite, two separate frameworks.

So whenever you see AngularJS that's basically legacy work. Current work is Angular 2 and higher.

# TypeScript

TypeScript provides static typing support to JavaScript .It is a superset of JavaScript and ECMAScript.

TypeScript is a strongly-typed language with compile-time checking and IDE support.

Web browsers do not understand TypeScript natively, they can only understand JavaScript, so we somehow need to convert our TypeScript code over to JavaScript code. And this is known as transpiling.

tsc mydemo.ts --> will convert mydemo to mydemo.js to run the javascript file node mydemo.js

So we're using the new typescript 'let' keyword for variable declarations, as opposed to the traditional JavaScript var keyword.The JavaScript var keyword had a number of gotchas and pitfalls, such as scoping issues, capturing, shadowing, and so on.

tsc --init will generate a new tsconfig.json file , where we can configure the workspace ts files.

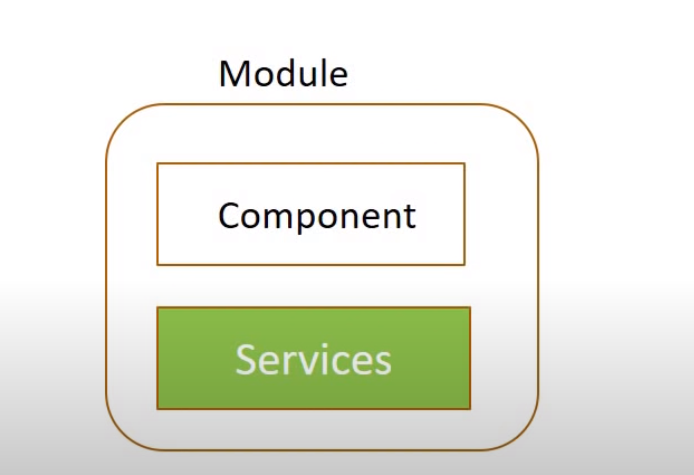
# Angular Architechture

## Module

An angular application is a collection of many individual modules , every module represents the feature area in the application.

It can be imported and exported.

Every angular application has atleast one module which is the root module - **AppModule**



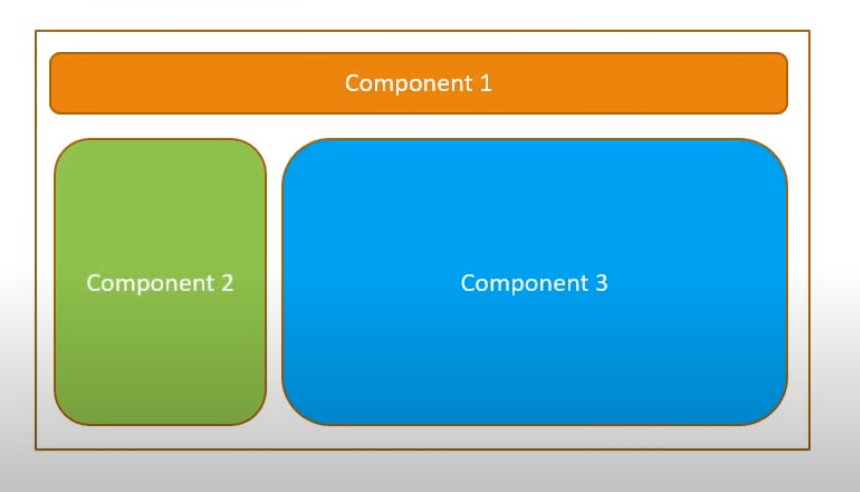
## Component

Each module has been made of components and services.

A component controlls a portion of the view on the browser. (for ex : we can have a component for navigation , side bar and main content )

Every angular application has atleast one component which is the root component – **AppComponent**

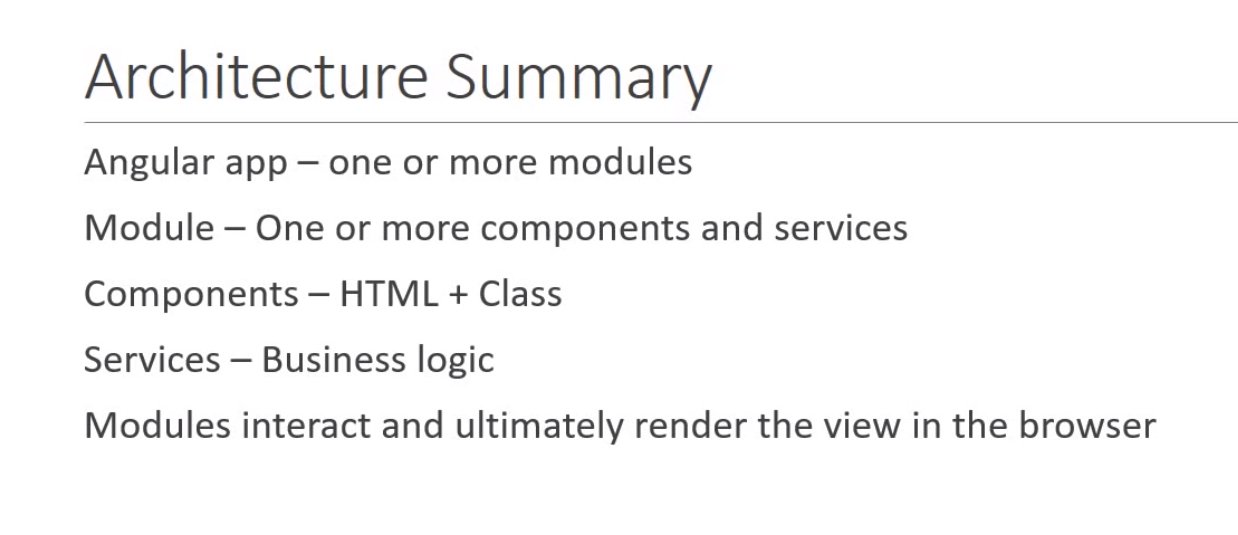
All other components will be nested under this **AppComponent.**



Each component will have a HTML template to represent the view in the browser and a class that controls the logic of that particular view.

# Services

Modules will also have the service which is a class that contains the **business logic** of the application.



# Project structure

## angular.json

The angular.json file is a configuration file that is located at the root level of an Angular workspace.

It provides workspace-wide and project-specific configuration defaults for build and development tools.

Eg : The **projects** property is an object that contains a configuration object for each project in the workspace. Each project configuration can specify things like the project's root directory, source files, output paths, and build options.



## package. json

It has list of node dependencies. as being similar to Maven's pom.xml file

basically a list of all your dependencies for your project.

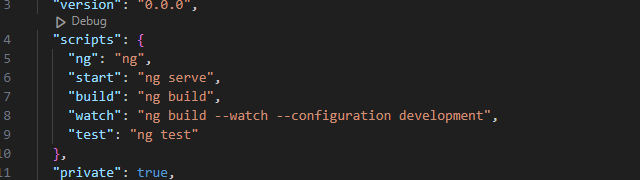
Eg :

The dependencies property is an object that lists all the packages that the project depends on, along with their version numbers.

When you run the npm install command, npm reads the dependencies property and installs all the packages listed there.



The scripts property is an object that lists all the scripts that can be run using the npm run command.

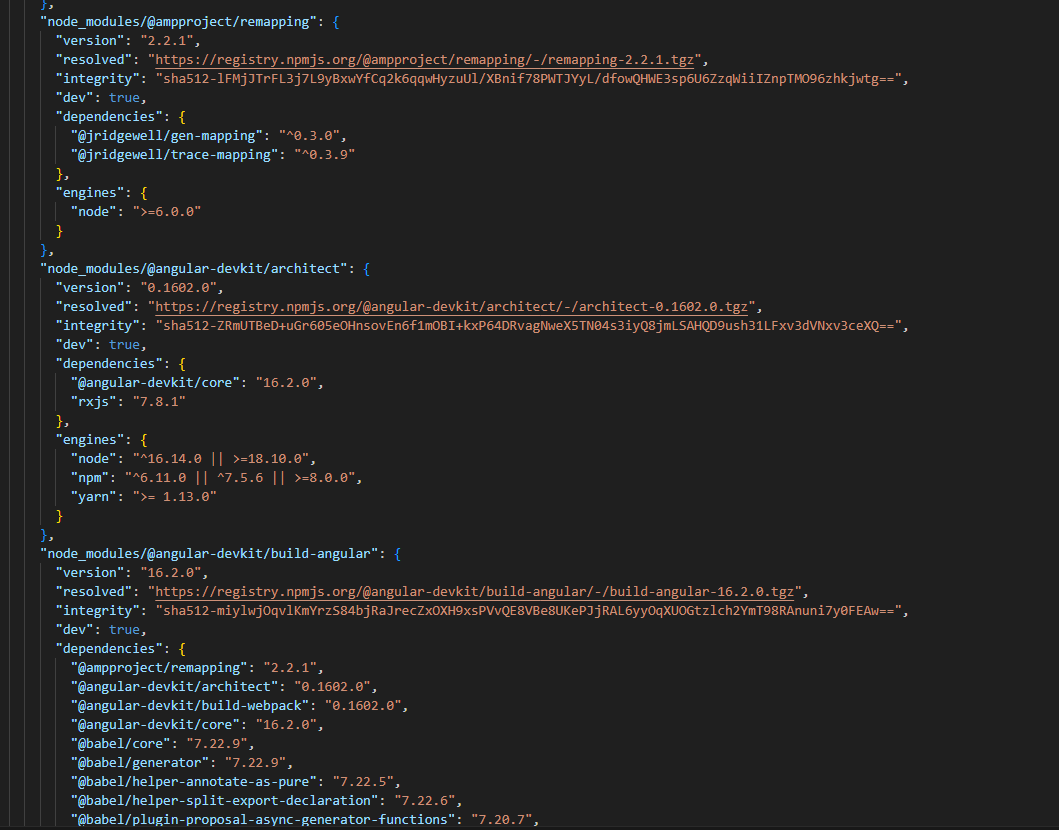


## package-lock.json

The package-lock.json file is a lockfile that holds information on the dependencies or packages installed for a Node.js project, including their exact version numbers.

It is automatically generated for any operations where npm modifies either the node\_modules tree or package.json

It ensures that the exact same version of the packages is installed every time and optimizes the installation process.



## src

The main source directory.

## app

the app folder basically has your app components, templates and so on.

## asset

The assets folder has any other web assets that you want to serve up. So for example, images and so on you can place here.

## environments

This folder has configuration so for like dev environment, test environment, prod environment,similar to profiles in the Spring world.

## index.html

This is your main launch page so this basically kind of bootstraps the entire process for your given application.

## polyfils.ts

Adds support for different browser versions for like i.e. Chrome, Firefox, Safari.

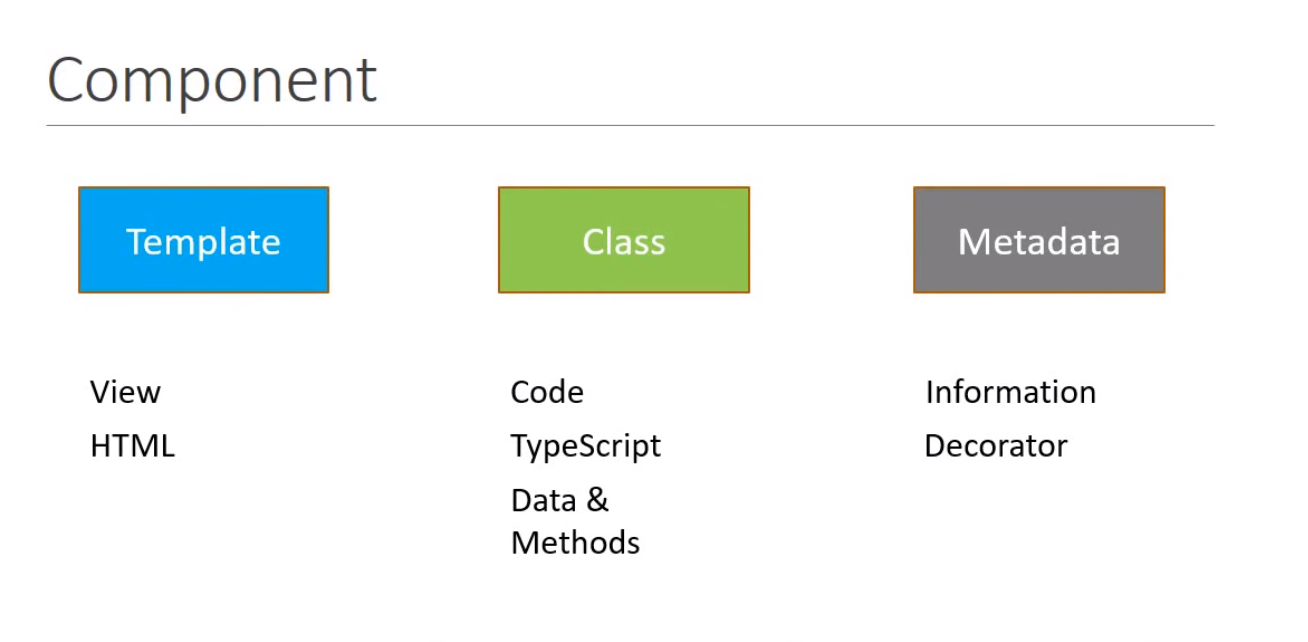
## test.ts

basically has a collection of unit tests for the entire application

## e2e , karma.config.js

This is for end to end testing.

# Component



A component is made up of 3 parts.

## Template

This is created using HTML , will be the user interface / view for the application.

## Class

This is the code that supports the view.This is created using TypeScript

It like any other class contains Data properties and Methods to control the logic of the View.

## MetaData

It is the information that is required to decide that a particular class is a angular component or just a class.Metadata is defined using a Decorator.

Eg : @Component

@Component({

  selector: 'app-root',

  templateUrl: './app.component.html',

  styleUrls: ['./app.component.css']

})

Any time we create a new component , your application should be aware of it.

So , in the **app.module.ts** file , we will place it under declarations array.

