end to end tests by protractor (End to End (e2e))

├── e2e

Where testing code resides

│ ├── src

│ │ ├── app.e2e-spec.ts

 page object (Changes according to html code

the tests holder

│ │ ├── app.po.ts

Protractor how to set up the Selenium test

│ ├── protractor.conf.js

tsconfig.e2e.json*is the configuration file for testing*

│ ├── tsconfig.e2e.json

├── node\_modules

external dependencies

where our application lives

├── src

creates the root component, a root module, a unit test class

│ ├── app

routing required for your project

│ │ ├── app-routing.module.ts

CSS code will be available

│ │ ├── app.component.css

html code will be available

│ │ ├── app.component.html

automatically generated files which contain unit tests for source appappcomponen

│ │ ├── app.component.spec.ts

│ │ ├── app.component.ts

processing of the html structure

reference to different libraries of CLI

│ │ ├── app.module.ts

│ ├── assets

save your images, js files in this folder

│ │ ├── .gitkeep

production or the dev environment.

│ ├── environments

│ │ ├── environment.prod.ts

│ │ ├── environment.ts

Favicons generator

│ ├── favicon.ico

display the details from app.component.html

│ ├── index.html

importing the basic modules and where we start our project development

style file required for the project

│ ├── main.ts

mainly used for backward compatibility

│ ├── polyfills.ts

│ ├── styles.css

the unit test cases for testing the project

│ ├── test.ts

├── .editorconfig

 configuration file for the Visual Studio code

├── .gitignore

 files are not committed to git source

├── angular.json

configuration file for Angular CLI

│── browserslist

[compatibility of the app with different browsers](ttps://angular.io/guide/build#configuring-browser-compatibility)

├── karma.conf.js

 Configuration file for the [karma test runner](https://karma-runner.github.io/).

├── package-lock.json

 representation of a dependency tree

├── package.json

[package.json is an npm configuration file](https://docs.npmjs.com/creating-a-package-json-file),

├── README.md

 The Read me file

 used during compilation, it has the config details that need to be used to run the application

├── tsconfig.app.json

contains options and flags that are essential for Angular applications

├── tsconfig.json

maintain the details for testing

├── tsconfig.spec.json

TSLint is an extensible static analysis tool that checks TypeScript code for readability, maintainability, and functionality errors

└── tslint.json

**app folder**

**The Component**

* The app.component.ts is the component that is added to the project by Angular CLI.

You will find it under the folder app/src

 used to import the libraries that are used in our component

@Component (called class decorator) provides Metadata about our component

 the path to the HTML template file.



The component is a simple class. We define it using the export keyword. The other parts of the app can import it use it. The above component class has one property title. This title is displayed, when the application is run.

The component class can have many methods and properties. The main purpose of the component is to supply logic to our view

where to display the template

array of Style Sheets that Angular uses to style our HTML file

**Template**

* The app.component.html is our template.The templateUrl in the component class above points to this class.
* The app.component.html file is in the same folder as the AppComponent. The code is almost 500 lines of code and almost all of it is standard HTML & CSS
* Note that {{title}} is placed inside the h1 tags in line no 344. The double curly braces are the angular way of telling our app to read the title property from the component (AppComponent). We call this data binding (interpolation).

**Root Module**

* Angular organizes the application code as [Angular modules](https://www.tektutorialshub.com/angular/angular-modules/). The Modules are closely related blocks of code in functionality. Every application must have at least one module.
* The Module, which loads first is the[root Module](https://www.tektutorialshub.com/angular/angular-modules/#root-module). This Module is our root module.
* The root module is called app.module.ts. (under src/app folder). It contains the following code

The import statement is used to import the Angular libraries required by AppModule. like NgModule &BrowserModule. We also need to import AppComponent, as we want to load AppComponent, when we load the AppModule

Declaration Metadata lists the components, directives & pipes that are part of this module.

The Angular Modules require a @ngModule decorator. @ngModule decorator passes the metadata about the module.

The @ngModule Metadata above has four fields. The declarations, imports, providers, & bootstrap

Providers are the services that are part of this module, which can be used by other modules

**Module class**

Similar to the component, the Module class is defined with the export keyword. Exporting the class ensures that you can use this module in other modules.

Imports Metadata tells the angular list of other modules used by this module. We are importing BrowserModule and AppRoutingModule.

The BrowserModule is the core angular module, which contains critical services, directives, and pipes, etc.



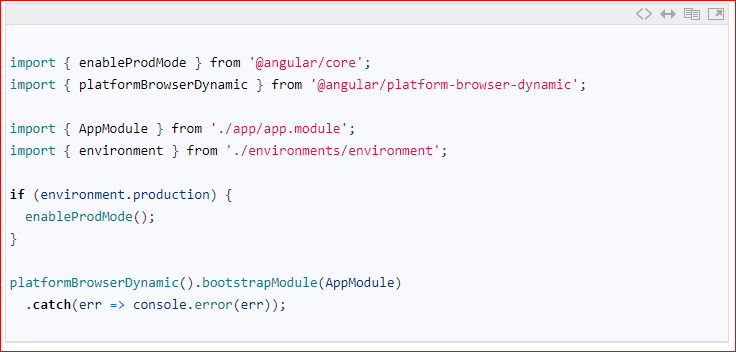
**App Routing Module**



* The AppRoutingModule in the file app-routing.module.ts defines the Routes of the application. These Routes tells Angular how to move from one part of the application to another part or one View to another View.
* The Routes defined in the constant const routes: Routes = [];, which is empty
* This Module is defined as a separate Module and is imported in  AppModule.

**Bootstrapping our root module**

import our AppModule



the bootstrapModule method of platformBrowserDynamic library to bootstrap our AppModule

our code checks the environmental variable and enable the production mode, if the environment is production

We also import is enableProdMode from @angular/core library. The Angular’s code by default runs in development mode.

The file environment.ts contains the contents for the current environment. The development environment uses the environment.ts file when you build

This library contains all the functions required to bootstrap the angular application.

* The app.component.html is the file, which we need to show it the user. It is bound to AppComponent component. We indicated that the AppComponent is to be bootstrapped when AppModule is loaded
* Now we need to ask the Angular to load the AppModule when the application is loaded. This is done in main.ts file
* The main.ts file is found under the src folder.

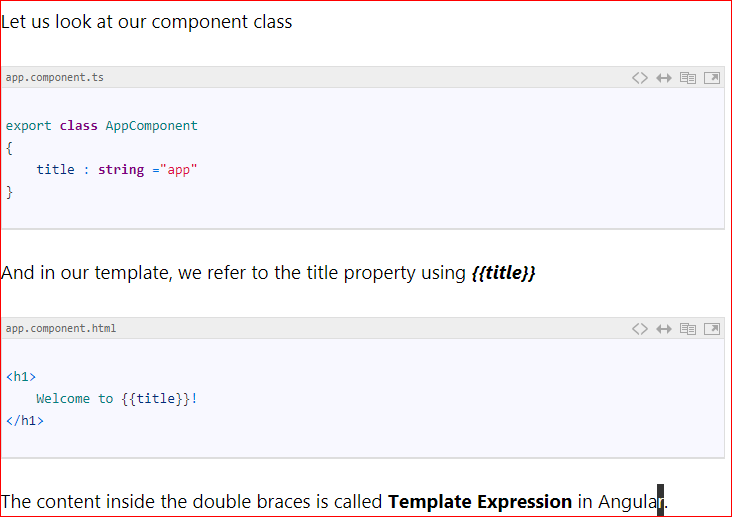
**Bootstrapping in Angular**

**What is a Bootstrapping?**

* Bootstrapping is a technique of initializing or loading our Angular application.
* let’s walk through  [First new Angular project](https://www.tektutorialshub.com/angular/angular-create-first-application/) and see what happens at each stage and how our AppComponent gets loaded and displays “app works!”. The Angular takes the following steps to load our first view.
* Index.html loads
* Angular, Third-party libraries & Application loads
* Main.ts the application entry point
* Root Module
* Root Component
* Template

**What is Angular Data Binding ?**

* Data Binding is a process where data is passed from Angular Component to view (Template) and vice versa. The Data Binding is used to bind DOM Elements to Component properties. Binding can be used display component class property values to the user, change element styles, respond to a user event etc.



* The Angular first evaluates the Template Expression and converts it into a string. Then it replaces Template expression with this result in the HTML Markup.
* The Template Expression is much more powerful than just getting the Property of the component class. You can use it to invoke any method on the component class or to do some mathematical operations etc.

### Template statement

* Template statement is similar to Template expression but can change the state of the application.
* Template statements are used in case of Event Bindings. It responds to the event raised by the user like clicking on a save button (Click event) or modifying the value of textbox (Change event) etc and invokes the method in the component class. Template statement is often is a method in the Component class
* Angular evaluates the Template statement just like the Template expression. Angular updates the view after the Template statement is invoked

### Difference between Template Expression and Template statement

* Template statement can change the Application state from the user input. The Template expression should not change the application state.
* Both use different Parsers.
* Template expression does not support assignment ( = only) operator. Template statement does.
* Template Expression does not support chaining of expressions. Template statement does.

**Data Binding in Angular**

There are four ways you can bind data in Angular

* **Interpolation**
* The title property, which is defined in the component class bound to the template using **double curly braces** in the template. This is called **Interpolation.**
* Interpolation provides the data-binding from **component to the View**. Interpolation is a one-way binding. It binds from Component Class to the Template.
* The Template expression (double curly braces ) used for interpolation in Angular. The Angular evaluates the Template expression and replaces it with the result.
* You can use interpolation to invoke a method in the component, Concatenate two string, perform some mathematical operations or change the property of the DOM element like colour etc.
* **Invoke a method in the component**
* We can invoke the components methods using interpolation. Open the app.component.ts and add the following function
  + getTitle(): string {
  + return this.title;
  + }
* Now open the app.component.html and copy the following

<p>{{getTitle()}}</p>

* **Concatenate two string**
  + **<p>{{ 'Hello & Welcome to '+ ' Angular Data binding '}}</p>**
* **Perform some mathematical operations**
  + **<p>{{100\*80}}</p>**

#### **Notes on Interpolation**

1. **Interpolation is one way from component to View**
2. **Binding source is a Template expression**
3. **Interpolation Expression must result in a string. If we return an object it will not work**

* **Property Binding**
* Property binding allows us to bind **Property of a view element** to the value of template expression
* **[Property]=”expression”**
* The Property Binding uses the **[] brackets**. The Binding Target is placed inside the square brackets. The Binding source is enclosed in quotes.
* Property binding is one way from Component to the Target in the template.

**Notes on Property Binding**

1. **Property Binding is one way from component to View**
2. **Binding source is a Template expression**
3. **Non-string return values are allowed**

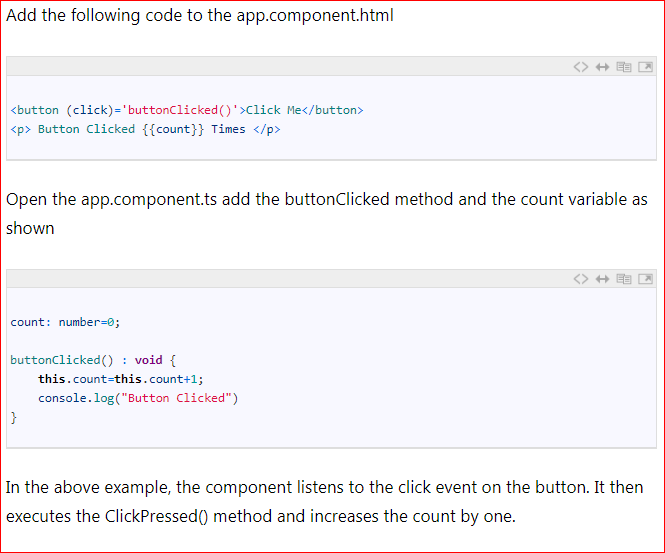
**Property Binding Vs Interpolation**

*Interpolation requires expression to return a string. If you want to set an element property to a non-string data value, you must use property binding.*

* **Event Binding**
* Event Binding is used to Perform an action in the component when the user clicks a button in the view.
* **For Example:**

**<button (click)=’ClickPressed()’>**

* The Event Binding uses the **parenthesis**. The Name of the event is enclosed in Parentheses. It is then assigned to the **Template Statement**. Template statement is often the method in the component class



#### **Notes on Event Binding**

1. **Event Binding is one way from View to Component**
2. **Binding source is a Template statement**



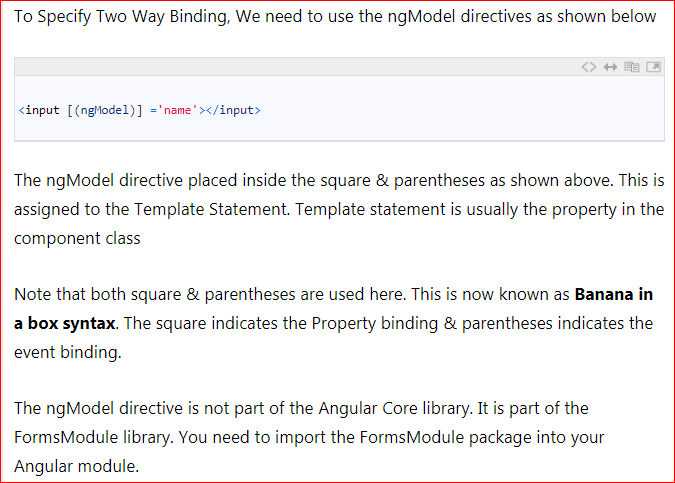
* **Two Way Binding**

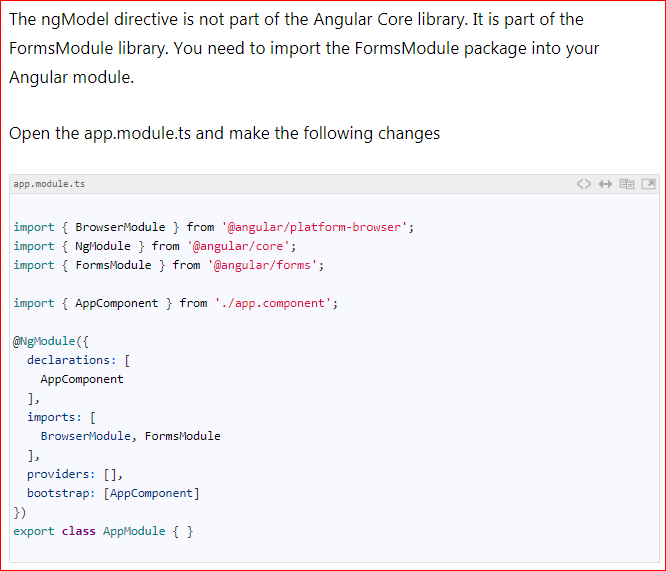
Two-way binding means that changes made in the component data are propagated to the view and that any changes made in the view are immediately updated in the underlying component data.

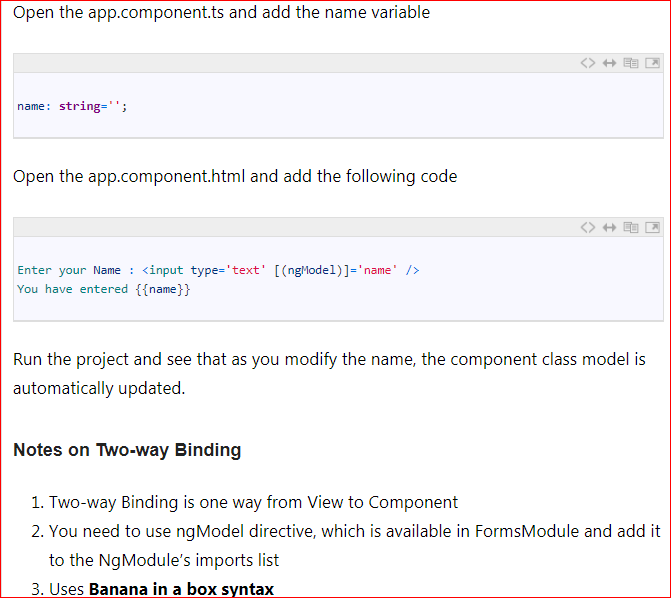
Two-way binding is used mainly in data entry forms. Whenever user makes changes in the data, we would like to update our model in the component with the new data and if the model changes, we would like to update the view as well

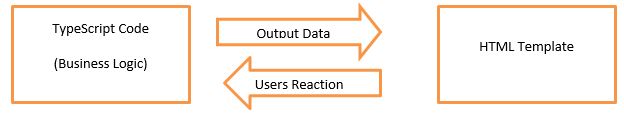
The Angular uses the combination of Property binding (from component to view) and event binding (from view to component) to achieve the Two-way data binding. This is done so by using the ngModel directive

To Specify Two Way Binding, We need to use the ngModel directives as shown below









**What is a Child/Nested Component**

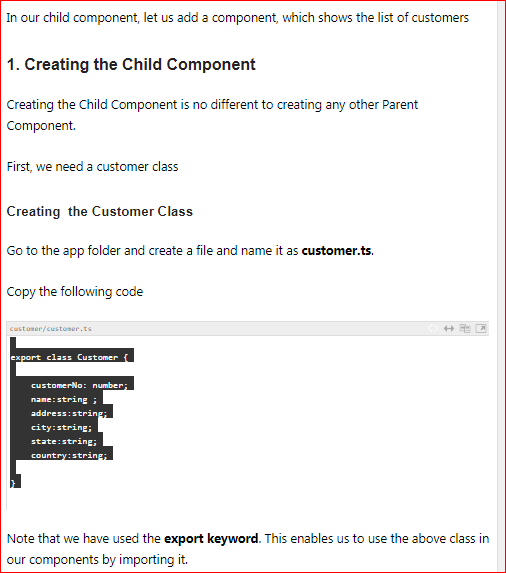
The Angular follows component based Architecture, where each component manages a specific task or workflow. Each component is an independent block of the reusable unit.

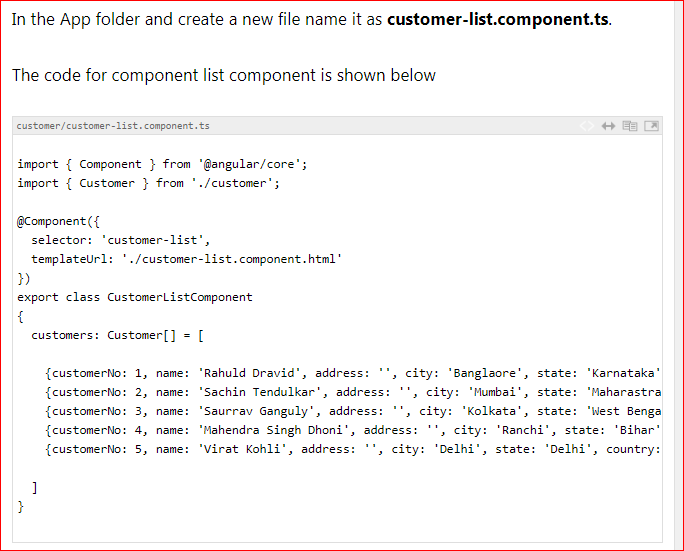
In real life angular application, we need to break our application into a small child or nested components. Then the task of root components is to just host these child components. These child components, in turn, can host the more child components creating a Tree like structure called Component Tree.

**Steps to Add child component:**

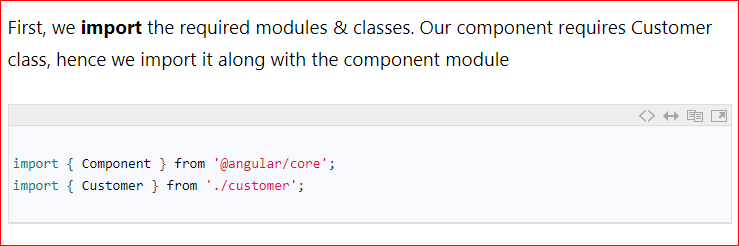
1. Create the Child Component. In the child Component, meta data specify the selector to be used
2. Import the Child Component in the module class and declare it in declaration Array
3. Use the CSS Selector to specify in the Parent Component Template, where you want to display the Child Component

**Adding a Child Component in Angular**

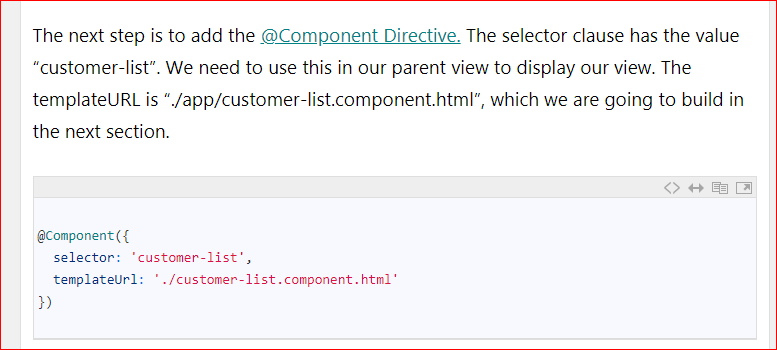




**Step 1:**



**Step 2:**

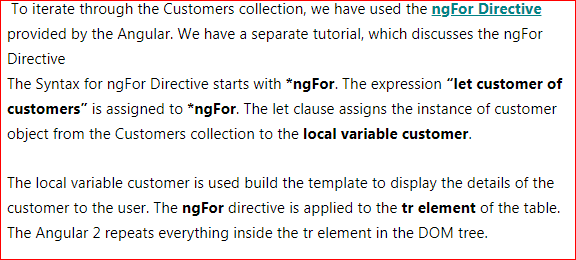


**Step 3:**

**Creating the View**

The next step is to create the view to display the list of customer. Go to the **app folder** and create the file with the name **customer-list.component.html**



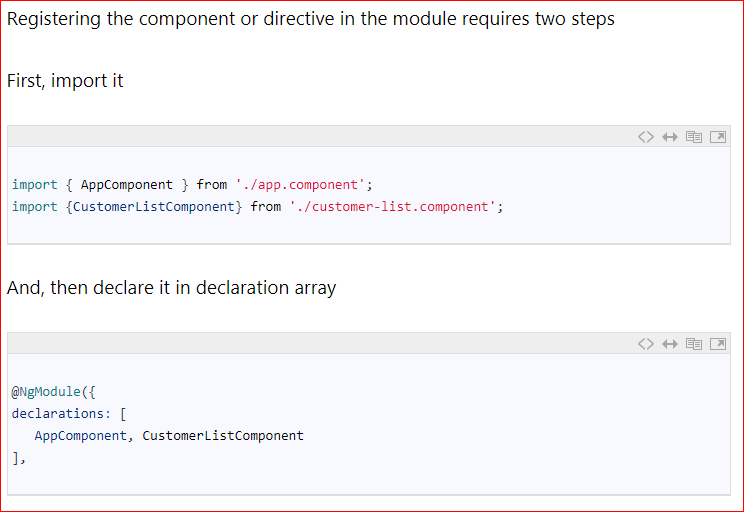


### 2. Import the Child Component in the Module

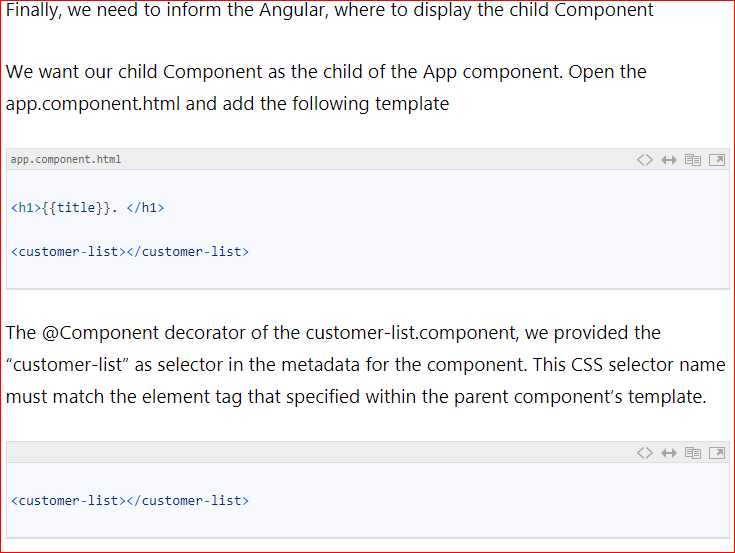
**Now, we have built the child component. We need to register it our app module.**

* The Modules (or NgModules) are Angular ways of organizing related components, directives and services etc into a Group. To add component to a module, you need to import it the Ngmodule file and declare it the declaration array
* Every component/directive that we build must be declared in the ngModule. A Component cannot be part of more than one module.
* The Module, which loads first is known as the root Module. Since our application has only one module (app.module), it will automatically become the root module
* Open **the app.module.ts** under the app folder and update the code as shown below





**3. Tell angular where to display our component**



**4.Run**

Run the application from the command line using **ng Serve**

**What is Angular Directive**

The Angular directive helps us to manipulate the DOM. You can change the appearance, behavior or layout of a DOM element using the Directives. They help you to extend HTML

There are three kinds of directives in Angular:

1. Component Directive
2. Structural directives
3. Attribute directives

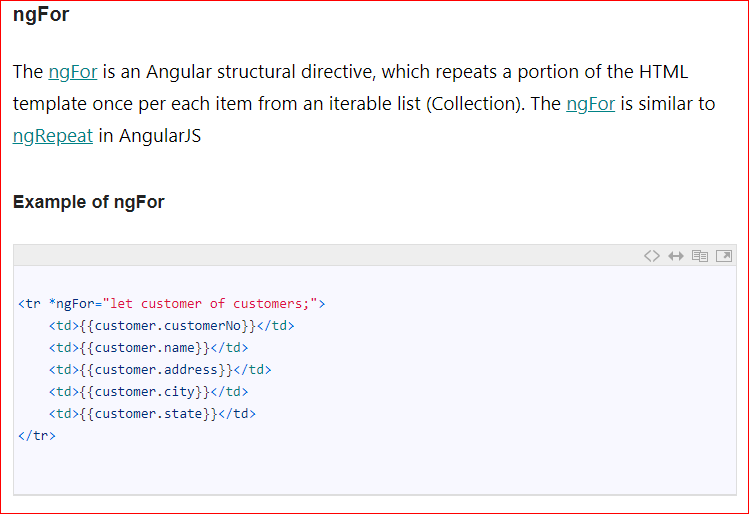
**Component Directive**

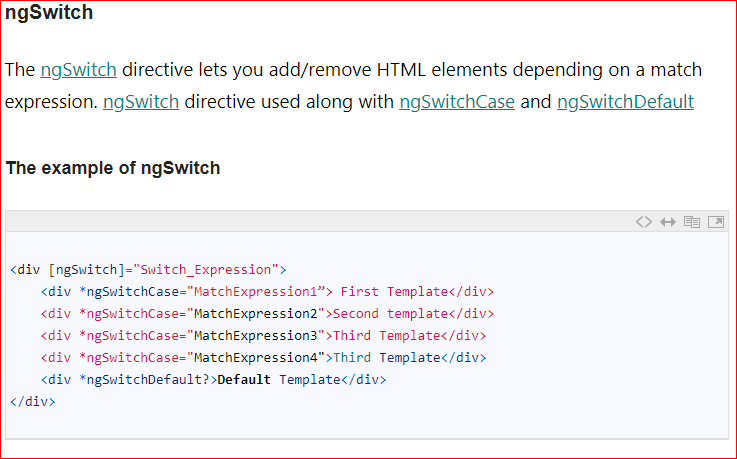
Components are special directives in Angular. They are the directive with a template (view).

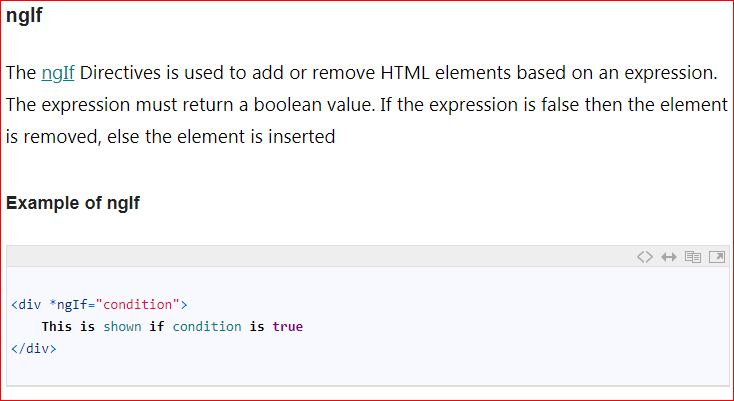
**Structural Directives**

Structural directives can change the DOM layout by adding and removing DOM elements. All structural Directives are preceded by Asterix symbol

**Commonly used structural directives :**





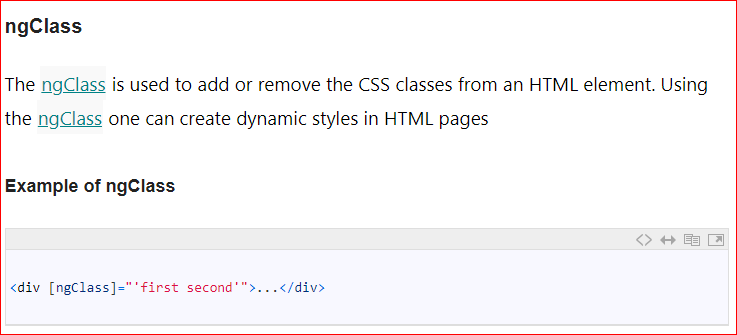
 **Attribute Directives**

An Attribute or style directive can change the appearance or behavior of an element.

Commonly used Attribute directives

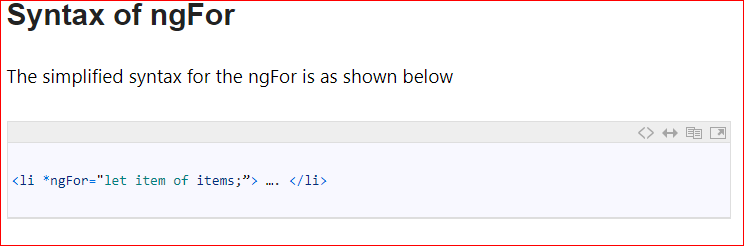
**ngModel**

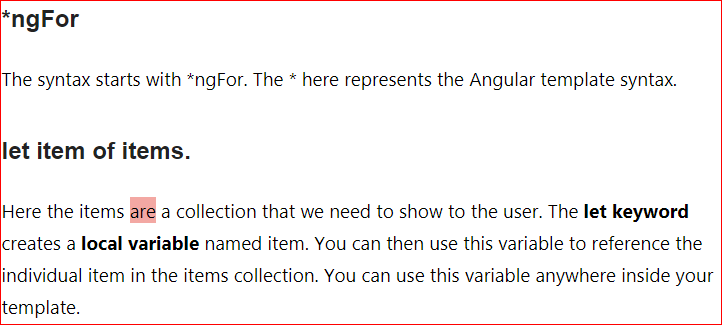
The ngModel directive is used the achieve the two-way data binding.

# **Angular ngFor Directive**

* The **ngFor is an Angular structural directive**, which repeats a portion of HTML template once per each item from an iterable list (Collection).





### Example of ngFor

### Step1:

### First, you need to create an angular Application.

### Step2:

### Create the movies-list.component.ts and add the following code. The following Code contains a list of Top 5 movies in a movies array. We will build a template to display these movies in a tabular form.

### Step3:

### import { Component } from '@angular/core';

### @Component({

### selector: 'movie-app',

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

### styleUrls:['./app/app.component.css']

### })

### export class AppComponent

### {

### title: string ="Top 10 Movies" ;

### movies: Movie[] =[

### 

### {title:'Zootopia',director:'Byron Howard, Rich Moore',cast:'Idris Elba, Ginnifer Goodwin, Jason Bateman',releaseDate:'March 4, 2016'},

### {title:'Batman v Superman: Dawn of Justice',director:'Zack Snyder',cast:'Ben Affleck, Henry Cavill, Amy Adams',releaseDate:'March 25, 2016'},

### {title:'Captain American: Civil War',director:'Anthony Russo, Joe Russo',cast:'Scarlett Johansson, Elizabeth Olsen, Chris Evans',releaseDate:'May 6, 2016'},

### {title:'X-Men: Apocalypse',director:'Bryan Singer',cast:'Jennifer Lawrence, Olivia Munn, Oscar Isaac',releaseDate:'May 27, 2016'},

### {title:'Warcraft',director:'Duncan Jones',cast:'Travis Fimmel, Robert Kazinsky, Ben Foster',releaseDate:'June 10, 2016'},

### {title:'Avengers: Age of Ultron',director:'Joss Whedon',cast:'Robert Downey Jr., Chris Evans, Mark Ruffalo',releaseDate:'May 1, 2015'}, ]}

### Step4:

### Create movies.ts file and paste for following code:

### class Movie {

### title : string;

### director : string;

### cast : string;

### releaseDate : string;

### }

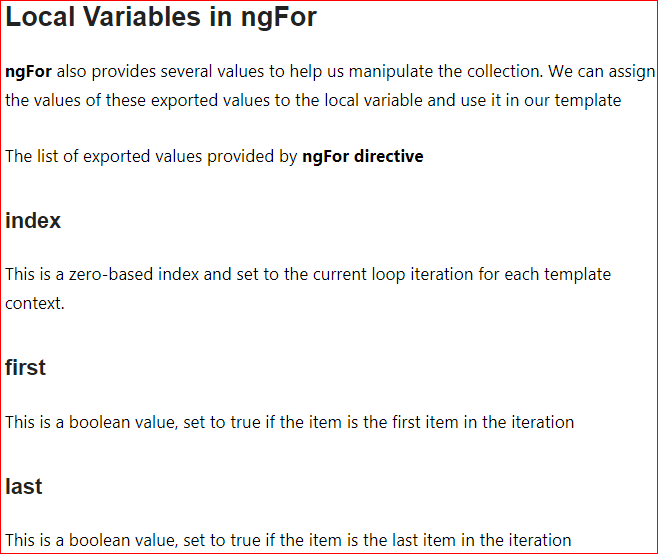
### Step5:

The next step is to create HTML template. Open the movies-list.component.html and add the following code:

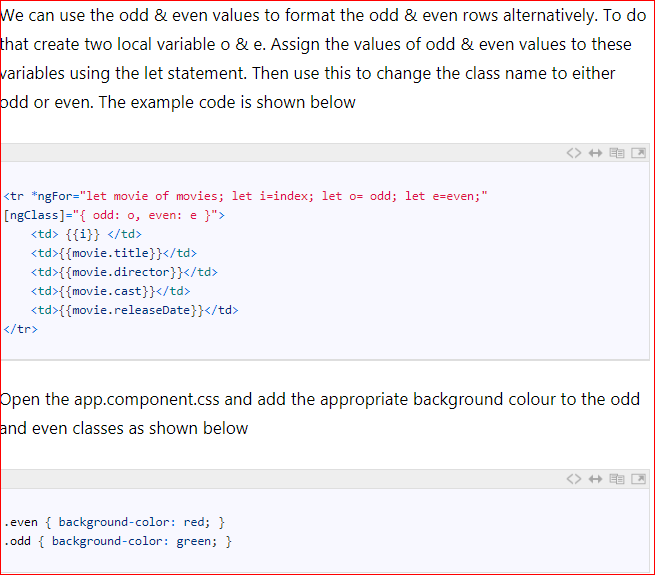


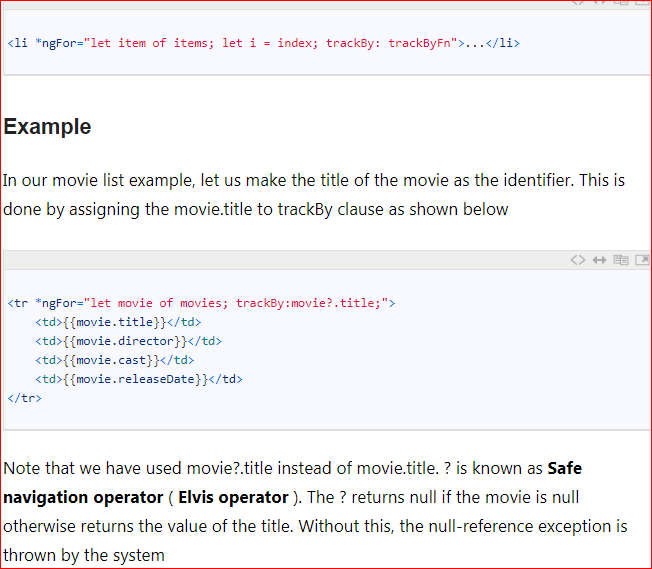
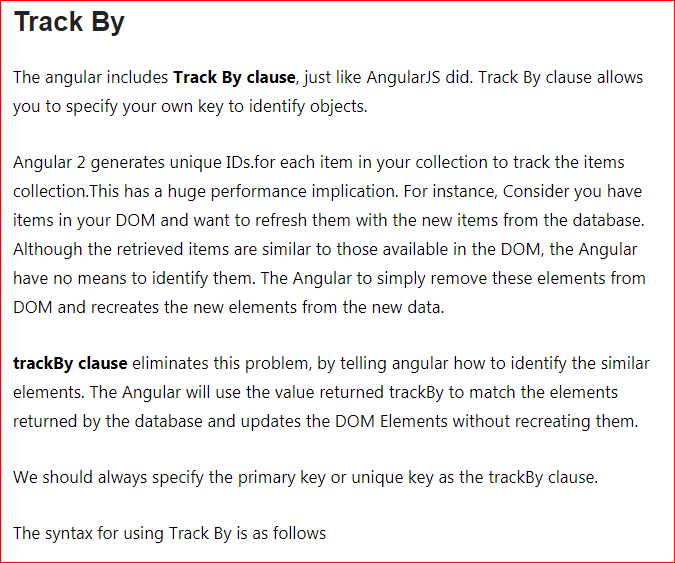
Our purpose is to display the list of movies. Hence we build a **tr element** of a table, which can display a single movie. We want the **tr element** to be repeated for each item in the movies collection. Hence we apply **ngFor directive to the tr element of the table**.

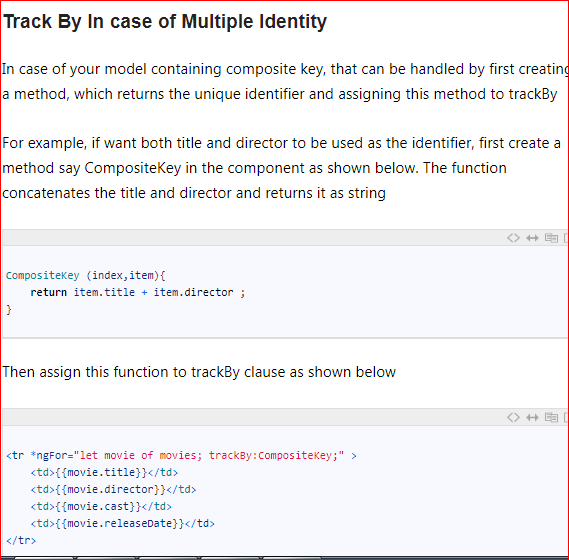
The **“let movie of movies”** creates the **local variable movie**. You can use the variable to create the template.



**Formatting odd & even rows :**

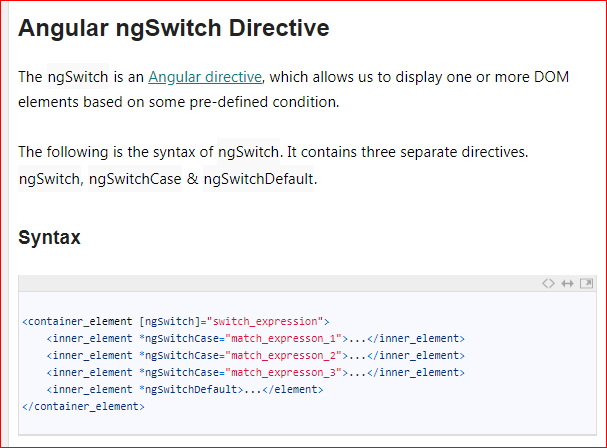


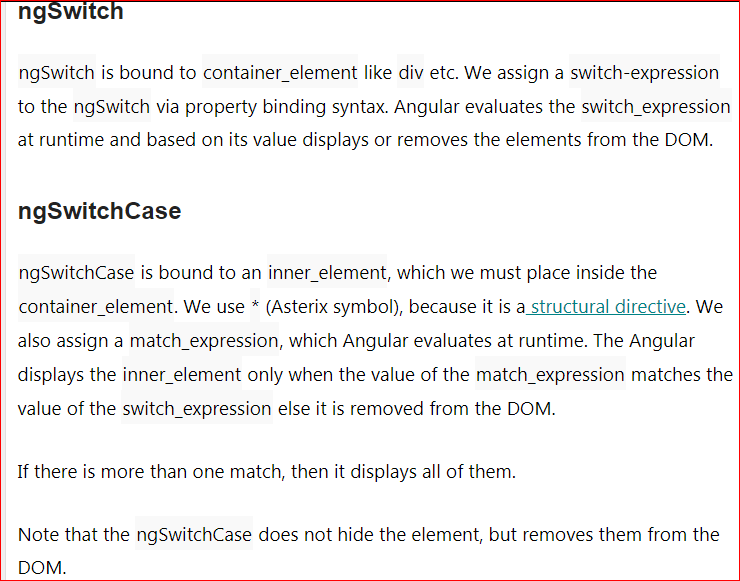
 

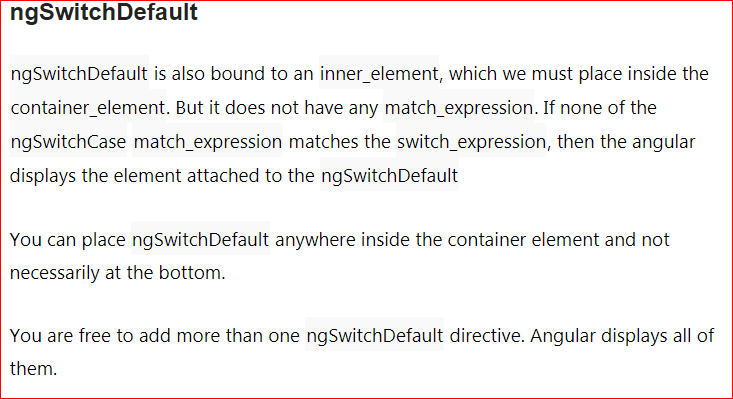


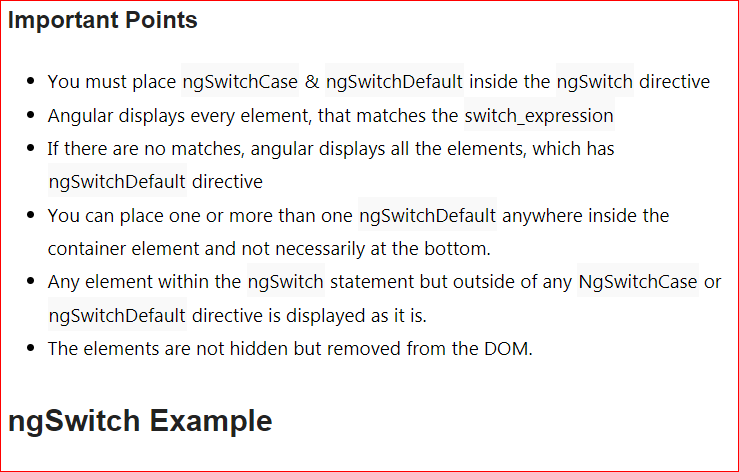
# **ngSwitch, ngSwitchcase, ngSwitchDefault Angular Example**

The ngSwitch is an Angular structural directive, which allows us to add or remove DOM elements. It works in conjunction with ngSwitchcase, & ngSwitchDefault directives. It is similar to the switch statement of JavaScript









Note : Other directives code in the folder (angular codes) in Laptop D: Drive

**Angular 2 Pipes**

There are many circumstances where we may have to change the appearance of the data before presenting it the user. The most common examples are dates. That is where Angular 2 Pipes comes handy. They take the data as input and transforms that data to get the desired output.

