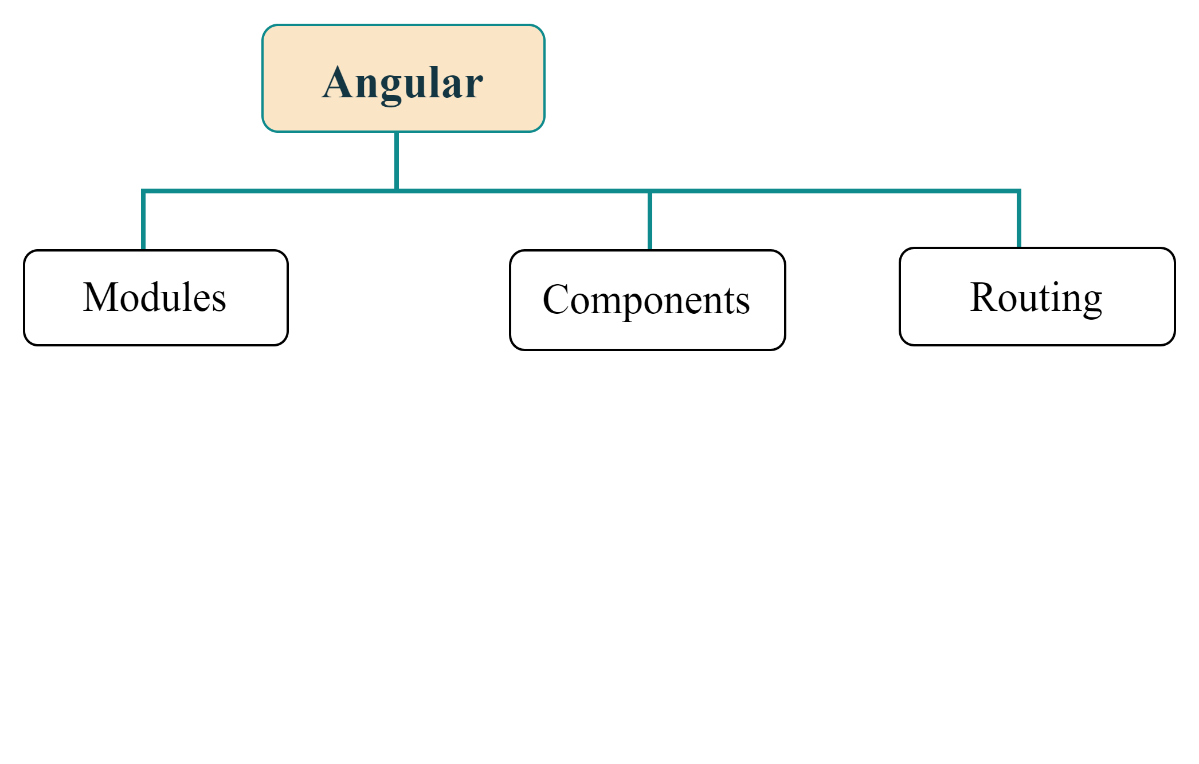
**ANGULAR**

Angular is a platform or framework to build client-based applications in HTML and TypeScript.

It is written in TypeScript.

Angular consists of three main things:



**Modules:**

Aka as ngModules.

modules are basic building blocks of angular applications.

for every application to have a root module in it.

**Components:**

Components are the basic building block of the user interface in Angular.

A component consists of HTML, CSS, and JavaScript for a specific portion of a user interface.

**The architecture of Angular comprises the following elements. The pictorial representation of the same is given below:**

* Components and Templates
* Ng Modules
* Metadata
* Data Binding
* Directives
* Services
* Dependency Injection

**Building blocks:**

* **Components**: A component can control numerous views wherein each of the views is a particular part on the screen. All Angular applications have a minimum of one component called the root component. This component is bootstrapped in the root module, the main module. All the components include the logic of the application that is defined in a class, while the main role of the class is to interact with the view using an API of functions and properties.
* **Data binding**: Data binding is the process in which the various sections of a template interact with the component. The binding markup needs to be added to the HTML template so that Angular can understand how it can connect with the component and template.
* **Dependency injection**: It uses DI so that it can offer the necessary dependencies, mainly services, to the new components. The constructor parameters of a component inform Angular regarding the numerous services needed by the component, and DI provides a solution that gives the necessary dependencies to the new class instances.
* [**Directives**](https://intellipaat.com/blog/tutorial/angularjs-tutorial/angularjs-directives/): Angular templates are of dynamic nature, and directives help Angular understand how it can transform the DOM while manifesting the template.
* **Metadata**: Classes have metadata attached to them with the help of decorators so that Angular will have an idea of processing the class.
* [**Modules**](https://intellipaat.com/blog/tutorial/angularjs-tutorial/angularjs-modules/)**:** Module or NgModule is a block of code organized using the necessary capabilities set, having one specific workflow. All Angular applications have at least one module, the root module, and most of the applications have numerous modules.
* **Routing**: Angular router helps interpret the URL of a browser to get a client-generated experience and view. This router is bound to page links so that Angular can go to the application view as soon as the user clicks on it.
* **Services**: Service is a vast category that ranges from functions and values to features that play a significant role in Angular applications.
* **Template**: The view of each component is linked with a template, and an Angular template is a type of HTML tag that allows Angular to get an idea of how it needs to render the component.

**Life cycle hooks:**

1. ngOnChanges() - Responds when Angular sets/resets data-bound input properties.
2. ngOnInit() - Initialize the directive/component after Angular first displays the data-bound properties and sets the directive/component's input properties/
3. ngDoCheck() - Detect and act upon changes that Angular can't or won't detect on its own.
4. ngAfterContentInit() - Responds after Angular projects external content into the component's view.
5. ngAfterContentChecked() - Respond after Angular checks the content projected into the component.
6. ngAfterViewInit() - Respond after Angular initializes the component's views and child views.
7. ngAfterViewChecked() - Respond after Angular checks the component's views and child views.
8. ngOnDestroy - Cleanup just before Angular destroys the directive/component.

Every Angular app contains a file named angular.json. This file contains all the configurations of the app. While building the app, the builder looks at this file to find the application's entry point. See the structure of the angular.json file:

1. "build": {
2. "builder": "@angular-devkit/build-angular:browser",
3. "options": {
4. "outputPath": "dist/angular-starter",
5. "index": "src/index.html",
6. "main": "src/main.ts",
7. "polyfills": "src/polyfills.ts",
8. "tsConfig": "tsconfig.app.json",
9. "aot": false,
10. "assets": [
11. "src/favicon.ico",
12. "src/assets"
13. ],
14. "styles": [
15. "./node\_modules/@angular/material/prebuilt-themes/deeppurple-amber.css",
16. "src/style.css"
17. ]
18. }
19. }

When the application enters the build section, the options object's main property defines the entry point of the application. The application's entry point is main.ts, which creates a browser environment for the application to run and calls a function called bootstrapModule, which bootstraps the application.

These two steps are performed in the following order inside the main.ts file:

1. **import** { platformBrowserDynamic } from '@angular/platform-browser-dynamic';
2. platformBrowserDynamic().bootstrapModule(AppModule)

In the above line of code, AppModule is getting bootstrapped.

The AppModule is declared in the app.module.ts file. This module contains declarations of all the components.

**Below is an example of app.module.ts file:**

1. **import** { BrowserModule } from '@angular/platform-browser';
2. **import** { NgModule } from '@angular/core';
3. **import** { AppComponent } from './app.component';
4. @NgModule({
5. declarations: [
6. AppComponent
7. ],
8. imports: [
9. BrowserModule
10. ],
11. providers: [],
12. entryComponents: [],
13. bootstrap: [AppComponent]
14. })
15. export **class** AppModule { }

In the above file, you can see that AppComponent is getting bootstrapped. It is defined in app.component.ts file. This file interacts with the webpage and serves data to it.

**Below is an example of app.component.ts file:**

1. **import** { Component } from '@angular/core';
2. @Component({
3. selector: 'app-root',
4. templateUrl: './app.component.html',
5. styleUrls: ['./app.component.css']
6. })
7. export **class** AppComponent {
8. title = 'angular';
9. }

Each component is declared with three properties:

1. **Selector -** It is used to access the component.
2. **Template/TemplateURL -** It contains HTML of the component.
3. **StylesURL -** It contains component-specific stylesheets.

Now, Angular calls the index.html file. This file consequently calls the root component that is app-root. The root component is defined in app.component.ts.

**See how the index.html file looks like:**

1. <!doctype html**>**
2. **<html** lang="en"**>**
3. **<head>**
4. **<meta** charset="utf-8"**>**
5. **<title>**Angular**</title>**
6. **<base** href="/"**>**
7. **<meta** name="viewport" content="width=device-width, initial-scale=1"**>**
8. **</head>**
9. **<body>**
10. **<app-root></app-root>**
11. **</body>**
12. **</html>**

The HTML template of the root component is displayed inside the <app-root> tags.This is the way how every angular application works.

**Coursera:**

Single Page Application (SPA):

Angular appln lives in a single HTML page. It lives all inside the <app-root> template.

Uses Angular Router for navigation.

Routes:

First define 🡪 register them with routerModule.forRoot() 🡪 add router outlet where router renders components 🡪 add router links to navigate to the defined Routes.

Resolvers:

Pre-fetching route data

Fetch the data in a resolve() method that’s invoked when the navigation starts

The router waits for the data to be resolved before the route is finally activated.

Ex: We can move the fetching of specific cocktail in a resolver. That way, we’ll have the cocktail when the details page is rendered and we won’t need a loading indicator.

Which Angular directive acts as a placeholder that Angular dynamically fills based on the current router state? RouterOutlet

Which of the following route definitions matches the path **pet-details** with parameter **name**?

const routes = [{

    path: 'pet-details/:name',

    component: PetDetailsComponent

}];

Which Angular directive, when applied to an element in a template, makes that element a link that initiates navigation to a route? RouterLink

Which Angular pipe subscribes to an Observable or Promise and returns the latest value it has emitted? async

Which Angular interface can classes implement to resolve data before a route is activated? Resolve

Which of the following templates will only render the **button** element if the **loggedIn** property is **true**?

Which Angular interface can be implemented by a class for guarding a route against unauthorized access? CanActivate

Which of the following is a route definition for lazy loading the **AdminModule**?

const routes = [{

  path: 'admin',

  loadChildren: () => import('./admin/admin.module')

    .then(m => m.AdminModule)

}];

Which interfaces can be implemented by a class to decide if the chunk for a lazy-loaded module should be fetched? CanLoad