Angular Home Page: <https://angular.io/>

Angular is developed by Google. Used 90% Typescripts applications and technology. NPM and Node JS is required to compile and run Angular.

**Local workstation setup:**

The following software is required to setup and install in local workstation to work on the Angular application.

1. Visual Studio 2019 or Visual Studio Code (2019 or latest version)

2. Angular CLI (latest version)

3. Node JS

4. Creates environment path variable for NPM. Use control panel => Edit Environment Variable for user account => update the below user path

C:\Users\\*\*UserNameORUserAccountId\*\*\appdata\Roaming\npm\

**IV Renderer:**

Angular extends HTML attributes with Directives, and binds data to HTML with Expressions

The ng-app directive defines an Angular application.

The ng-model directive binds the value of HTML controls (input, select, textarea) to application data.

The ng-bind directive binds application data to the HTML view.

**Angular Directives:**

The ng-init directive initializes Angular application variables.

ng-init

data-ng-init

<div ng-app="" ng-init="firstName='John'">

<div data-ng-app="" data-ng-init="firstName='John'">

We can use data-ng-, instead of ng-, if you want to make your page HTML valid

**Angular Expressions:**

Angular expressions are written inside double braces: {{ expression }}.

Angular will "output" data exactly where the expression is written:

<div ng-app="">

<p>My first expression: {{ 5 + 5 }}</p>

</div>

Angular expressions bind Angular data to HTML the same way as the ng-bind directive.

<div ng-app="">

<p>Name: <input type="text" ng-model="name"></p>

<p>{{name}}</p>

</div>

<div ng-app="" ng-init="myCol='lightblue'">

<input style="background-color:{{myCol}}" ng-model="myCol">

</div>

Like wise we can do Angular Numbers, Angular Strings, Angular Objects, Angular Arrays

<div ng-app="" ng-init="quantity=1;cost=5">

<p>Total in dollar: <span ng-bind="quantity \* cost"></span></p>

</div>

<div ng-app="" ng-init="firstName='John';lastName='Doe'">

<p>The name is <span ng-bind="firstName + ' ' + lastName"></span></p>

</div>

<div ng-app="" ng-init="person={firstName:'John',lastName:'Doe'}">

<p>The name is <span ng-bind="person.lastName"></span></p>

</div>

<div ng-app="" ng-init="points=[1,15,19,2,40]">

<p>The third result is <span ng-bind="points[2]"></span></p>

</div>

Angular Expressions vs. JavaScript Expressions

Like JavaScript expressions, Angular expressions can contain literals, operators, and variables.

Unlike JavaScript expressions, Angular expressions can be written inside HTML.

Angular expressions do not support conditionals, loops, and exceptions, while JavaScript expressions do.

Angular expressions support filters, while JavaScript expressions do not.

**Angular Applications:**

Angular modules define Angular applications.

Angular controllers control Angular applications.

The ng-app directive defines the application; the ng-controller directive defines the controller.

<div ng-app="myApp" ng-controller="myCtrl">

First Name: <input type="text" ng-model="firstName"><br>

Last Name: <input type="text" ng-model="lastName"><br>

<br>

Full Name: {{firstName + " " + lastName}}

</div>

<script>

var app = angular.module('myApp', []);

app.controller('myCtrl', function($scope) {

$scope.firstName= "John";

$scope.lastName= "Doe";

});

</script>

**Angular Modules:**

An Angular module defines an application.

The module is a container for the different parts of an application.

The module is a container for the application controllers.

Controllers always belong to a module.

<div ng-app="myApp">...</div>

<script>

var app = angular.module("myApp", []);

</script>

The "myApp" parameter refers to an HTML element in which the application will run.

Now you can add controllers, directives, filters, and more, to your Angular application.

Adding a Controller

Add a controller to your application, and refer to the controller with the ng-controller directive:

<div ng-app="myApp" ng-controller="myCtrl">

{{ firstName + " " + lastName }}

</div>

<script>

var app = angular.module("myApp", []);

app.controller("myCtrl", function($scope) {

$scope.firstName = "John";

$scope.lastName = "Doe";

});

</script>

**Adding a Directive:**

Angular has a set of built-in directives which you can use to add functionality to your application.

https://www.w3schools.com/angular/angular\_ref\_directives.asp

<div ng-app="myApp" w3-test-directive></div>

<script>

var app = angular.module("myApp", []);

app.directive("w3TestDirective", function() {

return {

template : "I was made in a directive constructor!"

};

});

</script>

<!DOCTYPE html>

<html>

<script src="https://ajax.googleapis.com/ajax/libs/Angular/1.6.9/angular.min.js"></script>

<body>

<div ng-app="myApp" ng-controller="myCtrl">

{{ firstName + " " + lastName }}

</div>

<script src="myApp.js"></script>

<script src="myCtrl.js"></script>

</body>

</html>

Functions can pollute the Global Namespace

Global functions should be avoided in JavaScript. They can easily be overwritten or destroyed by other scripts. Angular modules reduce this problem, by keeping all functions local to the module.

**When to Load the Library:**

While it is common in HTML applications to place scripts at the end of the <body> element, It is recommended that you load the Angular library either in the <head> or at the start of the <body>.

This is because calls to Angular. Module can only be compiled after the library has been loaded.

https://www.w3schools.com/angular/angular\_modules.asp

https://en.wikipedia.org/wiki/Angular

**Steps to work Angular JS:**

**Angular commands for usage:**

|  |  |  |
| --- | --- | --- |
| # | Command | Usage |
| 1 | >npm –v | Find specific npm version |
| 2 | >ng new AppName | Create new apps using Angular CLI command  Example: >ng new MyApp |
| 3 | TS | TypeScript file. JS and extra feature in JS. Superset of JS and backward compatibility |
| 4 | >npm i –g TypeScript | Install TypeScript |
| 5 | >tsc –version | Find TypeScript version |
| 6 | >tsc | TypeScript compiler. It will convert the \*.ts files into \*.js files. Also generates JS methods based on the method defined in TS |
| 7 | >npm i @angular/cli -g | G denote globally. Angular dependencies will install globally |
| 8 | >npm serve –o | Start and serve Angular apps |
| 9 | >ng build | Start build the Angular apps |
| 10 | >ng test | Start testing the Angular apps |
| 11 | Package.json | 2 types of dependencies Dev Dependencies and Dependencies. Dev Dependencies for local workstation to build the apps. Dependencies for deployment perspective. NPM is going to manage our apps |
| 12 | protractor, karma | Test the apps |
| 13 | rxjs | reactor programming in js |
| 14 | node\_modules folder | If we are request ng server to start in Angular CLI commands. It will take the dev dependencies files from here to validate, manage the app beautiful.  All libraries are downloaded here when we do npm install in first time  Packages specified in package.json file are installed into this folder |
| 15 | >npm i jquery | If we do then jquery will be downloaded from NPM repo and put it in node\_modules |
| 16 | >ng new app name | Creating new apps |
| 17 | app.module.ts | App module is core module.  Relative path is required for app module.  relative path is not required if we are referring from node\_modules |
| 18 | @NgModule | ngmodule is decorator. We can use to mention the components.  app.component.ts  selector => app-root. Check index.html it will refer app-root.  templateUrl => path for HTML file  styleUrls => path for CSS file |
| 19 | Components | components like shows in UI  app-root - whenever the app request comes here it will search for components. The compiler can understand the request and navigate.  app.component.html  Whenever we do changes and automatically saved and created junk files.    Binding, Editing  Angular is two way data binding flow between view to controller and controller to view.  for data binding in angular JS  {{variablename}}  Pipe  {{variablename|uppercase}} = it will transform into uppercase  we can do calculation as well  {{2\*2}} |
| 20 | > ng g directive default-directive-example | Create directive |
| 21 | >ng destroy directive default-directive-example | Destroy directive |
| 22 | >ng g component directive-example | Create component |
| 23 | Directives | structural directives => default directives  attribute directives => default directives  custom directives => own directives to create structural or attribute directives  ng if - angular if. We can use within html tags.  <p \*ngif="showContent"> show content based on ng if directive variable output</p>  The value of showContent is true.  Set of function which we can apply html based on the directives.  showContent: boolean = true  for loop in html  <p \*ngFor="let product of data">options are {{product}}</p>  we have to create a product list in model  data: Array<any> = ['OnePlus','nokia','Apple','Samsung']  employeeDetails : Array<object> = [{  {name : 'Balaji',  id : 1,  skills : 'Azure'},  {name : 'Govindarajan',  id : 2,  skills : 'AWS'},  }]  declaration part  <tr \*ngFor='let employeeDetails'>    i/p and o/p decorators  reactive forms  routing  lazy loading  promises, observables, REST API concepts |
|  | NPM | Node Package Manager |
|  | >ng g c ComponentName | Create new component |
|  | >ng build –dev | Development build  DEV build feature  Source map is available only for Dev. Source Maps helps us easily debug our  Application even after the files are compressed and combined.  >ng build --dev -sm true |
|  | >ng build –prod | Production build  PROD will do the below things  Tree Shaking => Remove the unused codes from Angular  Minification, Uglification => Remove the whitespaces, Remove the variable length function name.    AOT compilation => pre-compilation |
|  | >ng serve | Serve pages. Check the dependencies and builds the Angular |
|  | >npm install | Install dependencies into node\_modules |
|  | >npm install source-map-explorer –saved | Source map is available for dev and prod builds but we have to enable it  Source map will draw a map and analyze the dependencies and give size of the NPM packages.    How to Install  >npm install source-map-explorer --saved  To validate  >node\_modules\.bin\source-map-explorer dist\vendor.bundle.js  vendor.bundle.js having all the application code once build. |
|  | JIT Vs AOT compilations | JIT - JIT is runtime compilation. It won't compile when we request build time  By default > ng build > ng serve use JIT compiler. AOT - pre-compiled the apps. we have to explicitly declare this keyword before request build  >ng build --aot true  >ng serve --aot true |
|  | Angular Files | Manually setting up an Angular application from scratch |
|  |  | 1. Create application folder and add the configuration files  2. Install the NPM packages  =>Use node package manager we have to install all the dependencies NPM packages  3. Create root application component    =>src\app\app.component.ts  Angular app bootstrap will start the app component first. by default the root component is called app component  4. Create root application module  =>src\app\app.module.ts  by default Root application modules is app module  5. Create main.ts file  =>Entry point for our application. Look at this code in main.ts will boot start the app module.  6. Create index.html  =>SPA page. The root component view template will insert in Index.html with the help of "Selector" tag (available in root component) |
|  | package.json | Contains packages to build and run our angular application. Custom scripts can also be added. Dev Dependencies only required building the app in local workstation. This is not required when we take production builds. Both dependencies and dev dependencies. NPM packages are deployed in node\_modules once NPM install start |
|  | tslint.json | Configuration file for linting |
|  | Angular-CLI.JSON | Angular workspace configuration file.  It contains project name for app, Root directory,  Output directory, Assets  like favicon, Assets, Application index page, polyfile.ts to run apps in any  browser, TS compiler configuration(tsconfig), TS test case compiler configuration  files(testTsconfig), App prefix (we can customize this one),App styles  (We can use ng command to use scss or css), Application environment files. |
|  | e2e Folder | Contains end-to-end tests and their configuration files |
|  | .editorconfig | Configuration file for Visual Studio Code |
|  | .gitignore | Files and folders listed in this file are ignored, when a change set is checked in to source control |
|  | karma.config.js | Configuration file for karma(Unit Test Runner) |
|  | protractor.config.js | Configuration file for Protractor(e2e Test Framework) |
|  | ReadMe.md | Readme file which contains the commonly used Angular CLI commands out of the box |
|  | tsconfig.json | TypeScript compiler configuration file |
|  | src folder | Contains the angular project source code-Components, templates, pipes, services, images, styles etc |
|  | assets | Contains the assets of the application like images  any anything else to be copied when you build your application |
|  | environments | Contains the development & production environment files |
|  | favicon.ico | Favorite icon for the application |
|  | index.html | The main HTML page that is served to the user |
|  | main.ts | The main entry point for the applications. This file  Contains the code to bootstrap the application root module. |
|  | polyfile.ts | Contains polyfills needed by Angular |
|  | test.ts | Main entry for unit tests |
|  | tsconfig.app.json | TypeScript compiler configuration for the Angular App |
|  | tsconfig.spec.json | TypeScript compiler configuration for the Unit Tests |
|  | typings.d.ts | TypeScript typing file.TypeScript editors leverage these type definition files to display type information |
|  | app.component | The root component(App Component) TypeScript,  {ts,html,css,spec.ts} HTML template, Style Sheet and Spec files |

**Boilerplate template:**

We have to write a code for Export class, Component, Import component from Angular

Example : app.component.ts (The below code is boilerplate code)

Code #1

import {Component} from '@angular/core'

@Component({

Selector: 'my-app',

providers:[],

template:

<div style="padding:5px">

Balaji Govindarajan

</div>

})

export class AppComponent

{

userText: string = "Balaji Govindarajan"

}

Angular CLI is overcome the above issue to write the code again and again for each scenarios.

Code #2

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

@Component

({

selector:"app-root",

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

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

})

export class AppComponent

{

title ='app';

}

Selector => Match the app-root and bind the component

templateUrl => Contains html design or html reference file and path

styleUrls => Style sheet file and path

**Angular CLI:**

Angular CLI is command line tool the help us

>Create Angular applications faster and with great consistency

>Create the boiler plate code for Angular features like components, directives, pipes, services

>Create boiler plate code for TypeScript feature like classes, interfaces, enums

>It follows Angular best practices and conventions out of the box

>Run unit test cases and End-to-End(e2e) tests

>Create optimized builds for deployment to production

Software pre-request for Angular:

Node, NPM

Check for version use the below commands

>node -v

>npm -v

Install angular cli globally in machine

>npm install -g@angular/cli

check angular cli

>ng -v

If you into problems installing AngularCLI

1. Delete "npm" folder from "C:\Users\\*\*\*\*\*\*\*\AppData\Roaming"

2. Uninstall and reinstall node.js

3. Install AngularCLI again

**Use Angular CLI to create a new project:**

g => Generate

C => Component

ng => It stands for Angular

d =>dry run. it will generate any files but it will show the files

>ng new MyFirstApp

ng is angular cli itself. new is new project and app name. Everything is

created if we use this command.

>c:\MyFirstApp\code .

Code means to open visual studio code. dot - current path(MyFirstApp\)

>ng serve --open

Serve application and open the first page via default browser. by default

the port #4200 the app will run. If you want to change the port, we can

change it. this command build and launch the application.

live reload - means if we use same command, we can do the code and angular cli

will build the save changes (web server watch mode- detect the changes always)

and run the apps always.

>ng test

Run unit test. Build and run the unit test for application

>ng e2e

Test the application end to end

>ng --help

List out all the command for Angular. If we need to know any new command

>ng generate --help

Get specific command help

>ng --help | clip

Copy all the help information into windows clipboard

>ng --help > help.txt

Redirect all the output into help.txt notepad

>ng new --help

Creates new directory and a new Angular app.

>ng new -si -st

This is called alias(single dash). When we creating new app it will skip

to install NPM install and unit test cases

si - Skip installing packages

st - Skip creating tests

>ng new -si -st -it

it - Use inline templates when generating the new application

>ng new -si -st -is

is - Use inline styles when generating the new application

>ng -new app --dry-run

Run through without making any changes. Just reports the files that will be created

>ng -generate -component HomeScreen

it will generate new componnet in Angular Apps.

Example : c:\Apps\ng g c HomeScreen

>ng g c HomeScreen --style css

Angular will creates \*.scss (SASS CSS file). if we want to create \*.css then

we have to explicit declare the keyword like below.

>ng new App --prefix MyApp

we can explit declare the prefix for app selector.

>ng g c HomeScreen

Generate component for app

>ng g c HomeScreen/HallScreen

creates component another destinatio folder

>ng g c HomeScreen --flat

create a component without a folder

>ng g c MyApp -it -is

Create a component with an inline template and styles

>ng g c ghi --style=scss

Use SASS instead of CSS with your component

**Angular Service:**

>ng g s ServiceName

Generate Service in Angular. S is alias name for servicename

Example:

C:\MyAPp\ng g s customer

This command will create 2 files customer.service.spec.ts, customer.service.ts

We have to register service manually in root modules.

import { CustomerService } from './customer.service'

@NgModule

({

declarations: [AppComponent],

imports: [BrowserModule],

providers: [CustomerService],

bootstrap: [AppComponent]

})

>ng g s ServiceName -m=app.module

This command will register the service also along with file

Generations

>ng g s ServiceName --spec=false

this command will not generate the spec file. also we can use

-d on this command for dry run.

>ng g m ModuleName

generate the module for app. but we have register and imports

this module

>ng g m ModuleName -m=app.module

generate the module, register and imports the module

>ng g m ModuleName --spec=true

generate a specfile for the modules use --spec option

use - flat if you do not want folder

Generating Directives, Pipes and Guards

>ng g d directiveName

>ng g p pipeName

>ng g g guardName

Generating directives,pipes or components, when multiple exist

>ng g d directiveName --skip-import

>ng g d directiveName -module=app.module

also, we can use common ng generate command

--flat - dedicated folder should be created

--module - specifies the module with which the newly generated

angular feature should be registered

--spec - specifies if a spec file should be generated

Generating Classes, Interfaces and Enums

>ng g cl className

cl - classname

>ng g i interfaceName

>ng g e enumName

also, we can use folder to creat with in that

>ng g e foldername/enumname

Linting TypeScript

Linting checks for Programmatic errors, Stylistic errors,

Non-adherence to coding standards and conventions

>ng lint

Example : c:\MyApps\ng lint

tslint.json is used for declare linting rules for the apps.

>ng lint --fix

fix all the linting errors

>ng lint --type --check

Controls the type check for linting

--format - output format - json,stylish,verbose etc

Visual Studio Code extension TSLint is available to install and alert if any errors.

**Routes in Angular:**

If we need to routes for application then we need to install bootstrap package

>npm install bootstrap@3 --save

1. Set <base href="/"> in index.html

2. Create a separate routing module file (app-routing.module.ts)

3. Import the Angular RoutingModule in a separate module (app-routing.module.ts). Also don't forget to re-export RouterModule

4. Configure the application routes

5. Import the application routing module (AppRoutingModule) in the root AppModule

6. Specify where you want the routed component view template to be displayed using the

<router-outlet> directive

7. Create a navigation menu and tie the configured routes with the menu using the routerlink directive (we need to use bootstrap nav package for routelink)

Generating Routing Module using the AngularCLI

Angular CLI is providing generated templates for routing but

We have to do manually for configure the app components.

>ng new MyApp --routing

Will generate app with routing.

When we hit >ng serve --open Angular CLI will use webpacks to

Compile and build as bundles the apps in memory for web browser request

The ng serve command it won’t write disk files to serve application pages.

If you want to take DEV, UAT and PROD then need to use ng build

**Bundle File:**

inline.bundle.js => WebPack runtime.Required for WebPack to do it's job

main.bundle.js => All our application code that we write

polyfills.bundle.js => Browser Polyfills

styles.bundle.js => Styles used by the application

>ng serve - help

Get all the options for ng serve. Also use github page to see new things

https://github.com/angular/angular-cli/wiki/serve

>ng serve -o -w false

Switch off watch for file changes in Angular Apps.

>ng serve -o -w true -lr false

Switch off live reload function in web browser. The latest changes

builded but it won’t be available in browser

--watch - run build when files changes

--live-reload(lr)- whether to reload the page on change

--Open - Opens the url in default

--Port -- The port on which the server is listening

--extract(ec) -- Extract css into css file instead of js

>ng serve -o -p 8288

Change to diff. port

>ng serve -o -ec true

Extract the css during ng serve serve the pages.

>ng build

Building apps for development. it contains source maps files to debug the apps.

>ng build --dev

>ng build --prod

>ng serve

Compiles and serves the application from memory

Does not write the build files to the disk

Typically used to run the application on local development machine

Cannot be used for deploying the build to another server

(Ex. Testing, Staging or Production Server)

>ng build

Compiles the application to the "dist" folder

Can be used to produce both development and production builds

Typically used to deploy the application on another server

>ng build

or

>ng build --dev

With the DevBuild global styles are extracted to .js files whereas with the PROD build they are extracted to \*.css files

To generate a Dev Build with global styles extracted to \*.css file instead of \*.js ones

>ng build --dev -ec true