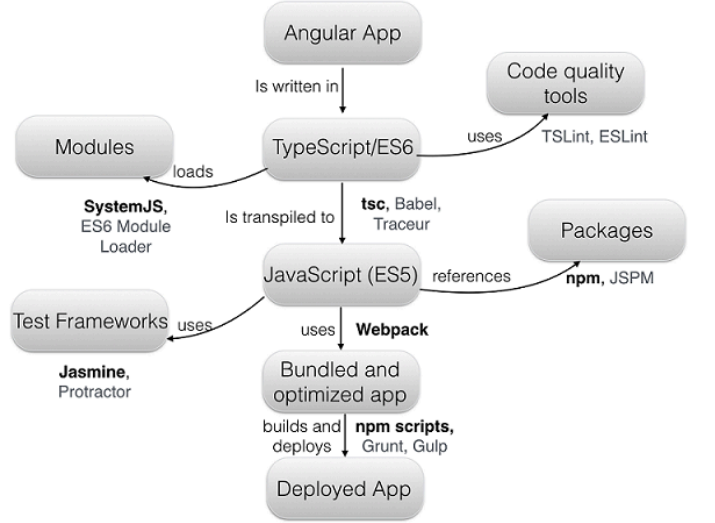
**ANGULARJS2**

**INTRODUCTION**

Angular 2 is an open source JavaScript framework maintained by google is used for developing web application. It’s a complete rewrite of its popular predecessor, AngularJS. Angular applications can be developed in JavaScript (using the syntax of ECMAScript 5 or 6) or Typescript

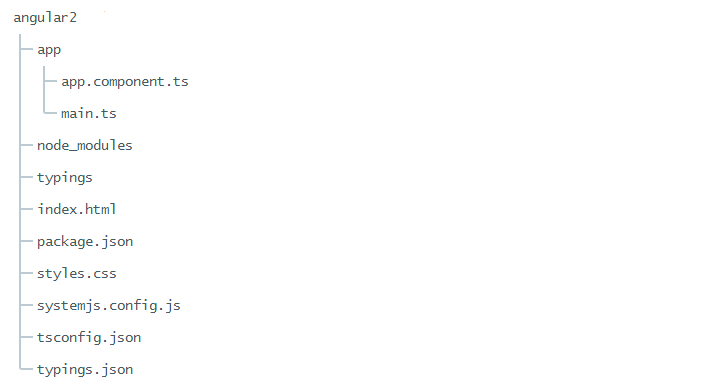


This is basics compilation procedure of angular2

***Why writing angular apps in typescript?***

* Typescript supports types. This allows the Typescript compiler help developers in finding and fixing lots of errors during development before even running the app.
* Typescript follows the ECMAScript 6 standard and adds to it types, interfaces, decorators, class member variables (fields), generics, and the keywords, and public  
  private.

**ANGULAR2-FOLDER STRUCTURE**



* **package.json** - lists packages the app depends on and defines some useful scripts
* **tsconfig.json -** is the Typescript compiler configuration file .It manages the version of translating to JavaScript file
* **typings.json -** identifies Typescript definition files. Many JavaScript libraries such as jQuery, the Jasmine testing library, and Angular itself, extend the JavaScript environment with features and syntax that the TypeScript compiler doesn't recognize natively. When the compiler doesn't recognize something, it throws an error. This JSON to tell the compiler about the libraries we load.
* **SystemJS-** SystemJS to load application and library modules.

**BUILDING BLOCKS OF AN ANGULAR APPLICATION**

**Components:**

Each component consists of two parts: a view that defines the user interface (UI) and a class that implements the logic behind the view.

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

@Component({

selector: 'my-app'

templateUrl: 'app/views/contact.html',

providers:[service name]

})

export class ContactComponent {

title = 'This is Contact Component';

}

 @Component configuration options:

* templateUrl: address of this component's template
* providers: array of **dependency injection providers** for services that the component requires.
* selector: CSS selector that tells Angular to create and insert an instance of this component

**DataBinding**

* {{title}} – displays the property value within the tag
* (click) [event binding](https://angular.io/docs/ts/latest/guide/user-input.html#click) calls the component's  method when the user click’s a button

**TwoWay DataBinding**

[(NgModel)] – Two way data binding

**Services:**

Services is used to passing value from one components to other components

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

@Injectable ()

export class SimpleService{

value = ‘’;

}

**@Injectable()-** marks a class as available to an injector for instantiation

**Component Router:**

The Angular Component Router enables navigation from one [view](https://angular.io/docs/ts/latest/guide/glossary.html#view) to the next as users perform application tasks.

import { provideRouter, RouterConfig } from '@angular/router';

import { HomeComponent} from './components/home';

import {ContactComponent} from './components/contact';

const routes: RouterConfig = [

{ path: 'home', component: HomeComponent },

{ path: 'contact', component: ContactComponent },

{ path: '', redirectTo: 'home'}

];

export const appRouterProviders = [

provideRouter(routes)

];

The RouterConfig is an array of routes that describe how to navigate. Each Route maps a URL path to a component.

**HTTP :**

Angular comes with its own HTTP library which we can use to call out to external APIs. Observables is used for asynchronous operation

For Post method:

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

Import {http} from ‘@angular/http’

@Component ({

selector: 'my-app'

templateUrl: 'app/views/contact.html',

})

export class ContactComponent {

Title = 'This is Contact Component';

Constructor (public http:http){

}

Makerequest():void{

let headers = new Headers({ 'Content-Type': 'application/json' });

let options = new RequestOptions({ headers: headers });

this.http.post(url, body, options).map((res: Response) => res.json());

}

}

When we bootstrap our app we will add HTTP\_PROVIDERS as a dependency. The effect is that we will be able to inject Http (and a few other modules) into our components.

**Main Component:**

import {bootstrap } from '@angular/platform-browser-dynamic';

import {AppComponent } from './js/components/boot';

import {appRouterProviders} from './js/route';

bootstrap(AppComponent, [

appRouterProviders

])

.catch(err => console.error(err));

**Bootstrap** : A method to configure and launch the root application component.

Providing the http providers at bootstrap makes the http service available everywhere in our application.

import { bootstrap } from '@angular/platform-browser-dynamic';

import {AppComponent} from './js/components/boot';

import { HTTP\_PROVIDERS } from '@angular/http';

bootstrap(AppComponent, [

HTTP\_PROVIDERS,DataService

]);

If we use shared service we inject into bootstrap makes data available everywhere in our application

**SOME VARIATIONS OF ANGULAR1 AND ANGULAR2 :**

|  |  |
| --- | --- |
| ng-app  this is application startup of angular1 | bootstrap(AppComponent);  Passing name of the component to bootstrap function |
| ng-click- for event binding | **(click)** -for event binding |
| **ng-hide-** hides the associated HTML | **[hidden]-** hides the associated HTML |
| ng-if | **\*ngIf** |
| ng-repeat | **\*ngFor** |
| ng-disabled | **[disabled]** |