* npm –v
* node –v
* npm install @angular/cli -g
* npm install -g typescript
* ng –v
* ng new my-new-project --style=scss –routing
* cd my-new-project

https://angular.io/guide/ajs-quick-reference

* Angular 2 is an open source JavaScript framework to build mobile & desktop web applications in HTML and JavaScript.
* Angular 2 is simpler than Angular 1. The concepts here are easier to understand.
* It was built by the developers at Google.
* Angular is itself written in Typescript.
* It will speed up the initial load through server side rendering.

**Features of Angular 2**

• Angular 2 is faster and easier than Angular 1.

• It supports the latest version of browsers and also supports old browsers including IE9+ and Android 4.1+.

• It is a cross-platform framework.

• Code structure is more simplified than the previous version of Angular.

**Advantages of Angular 2**

• If an application is heavy, then Angular 2 keeps it fully UI (User Interface) responsive.

• It works well with ECMAScript and other languages that compile with JavaScript.

• It uses dependency injection to maintain applications without writing lengthy codes.

* The applications here have a component-based approach.
* **Performance**

Faster Initial Load

Change Detection

Improve Rendering

Dependency Injection

Testability

* **Mobile Support: touch Interface and screen resolutions.**

**Key Features:**

* **Module**: Modules are used in Angular JS to put logical boundaries in your application. Hence, instead of coding everything into one application, you can instead build everything into separate modules to separate the functionality of your application.
* **Component:**  Components are a logical piece of code for Angular JS application. A Component consists of the Template, Class & Metadata.
* **Services** − Services are a set of code that can be shared by different components of an application.
* **Directive:** A directive is a custom HTML element that is used to extend the power of HTML. Angular 2 has the following directives that get called as part of the Browser Module module. Ngif & ngFor are built in directives.
* **Dependency injection:** Dependency injection is the ability to add the functionality of components at runtime.

**Difference Angular 1 & 2**

* Angular 2 is not an upgrade of Angular 1 but it is completely rewritten.
* Angular 2 provides more choice for languages. You can use any of the languages from ES5, ES6, typescript or Dart to write Angular 2 code while Angular 1.x had ES5, ES6, and Dart only.
* Angular 1.x was not built with mobile support in mind, where Angular 2 is mobile-oriented.
* Angular 1's core concept was $scope but you will not find $scope in Angular 2.0 and above.
* Angular 1.x Controllers are gone in Angular v2. We can say that Controllers are replaced with “Components” in Angular 2.
* In Angular 2, Structural directives syntax is changed. ng-repeat is replaced with \*ngFor.
* Angular 2 uses camel Case syntax for built-in directives. For example, ng-class is now ngClass and ng-model is now ngModel, ng-if = ngIf
* If you compare the file size, Angular 2 is 20 kb less than Angular 1 which helps in decreasing the load time for apps.
* One of the biggest advantages of Angular is Dependency Injection. In Angular 2, DI is there but now there is a different way to inject dependencies. As everything is a class in Angular, so DI is achieved via a constructor.
* In Angular 2, local variables are defined using hash(#) prefix.
* In Angular 1.x, we can define a service via 5 different ways.
  + Factory ,Service, Provider,
  + Angular 2, class is the only way to define a service.

**Angular 4**

* It is faster than Angular 2 with enhanced code compiling speed
* It is smaller than angular 2 with minimized the AOT (ahead-of-time) generated code. The bundles are reduced to thousands of KBs this change reduces 60% code generation for your components.
* new title case pipe for first character to make in capital
* Router ParamMap :
  + Till Angular 2, key-value object structure was used to store route parameters parameterObject[‘parameter-name’]
  + Angular 4, the parameters were made available in the form of a map & can be executed by simple method calls parameterMap.get(‘parameter-name’) type security. Old values were unsafe in regards to the type as these values could take any type possible. But now, these values are string or array of strings.
* Animation Package
* \*ngif with else introduced and ngFor use with as keyword

**Angular 5**

**Http Deprecated, HttpClient Here to Stay**

Before version 4.3, the @angular/http module was used for making HTTP requests in Angular applications. The Angular team has now deprecated Http in version 5. The HttpClient API from @angular/common/http package that shipped in version 4.3 is now recommended for use in all apps. The HttpClient API features include:

synchronous response body

Interceptors allow middleware logic to be inserted into the pipeline.

Material Design Components

Number, Date, and Currency Pipes

Support for Multiple Export Alias in Angular 5

@Component({

selector: 'app-root',

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

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

exportAs:'dashboard, logBoard'

})

Just-in-Time (JIT), which compiles your app in the browser at runtime

Ahead-of-Time (AOT), which compiles your app at build time.

If you are using angular2, the probability of you using TypeScript (TS) along with angular2 is very high. Since majority of the features of TS are still not supported by the browser, we might be relying on tools such as TypeScript compiler. These convert our code from TS to JS first, and then we serve these JS files to the browser (which again compiles the JS files to a binary which they understand).

**Typescript:**

* By definition, “Typescript is JavaScript for application-scale development.”
* Typescript is a strongly typed, object oriented, compiled language.
* It was designed by Anders Hejlsberg (designer of C#) at Microsoft.
* Typescript is a typed superset of JavaScript compiled to JavaScript.
* In other words, Typescript is JavaScript plus some additional features**.**
* At the end ts compiler convert ts file convert in to JS file.

**Features of Typescript:**

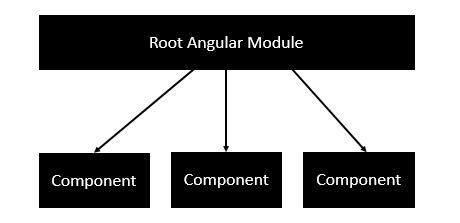
* JavaScript is typescript. This means that any valid .js file can be renamed to .ts and compiled with other typescripts files.
* Typescript supports other JS libraries
* A typescript is portable across browsers, devices, and operating systems. It can run on any environment that JavaScript runs on.
* Intellisense
* Auto Completion
* Code Navigation
* Strong Typing
* var num = 12; console.log(num);
* var num: number=12;

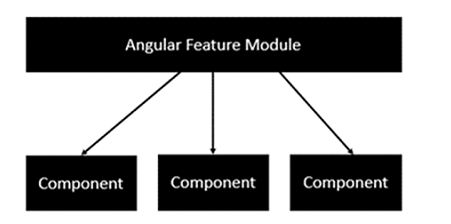
**Typescript Support several code editors like Visual studio, VS code, eclipse, etc...**

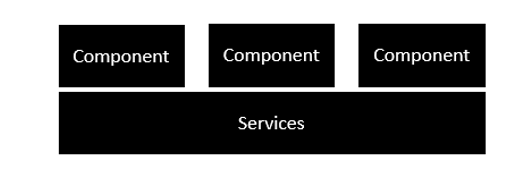
**EcmaScript European Computer Manufacturers Association Script**

* **JavaScript language standard official called ES.**
* ECMAScript is a subset of JavaScript. Languages such as Action Script, JavaScript, and JScript all use ECMAScript as its core.
* ECMAScript is commonly used for [client-side scripting](https://en.wikipedia.org/wiki/Client-side_scripting) on the [World Wide Web](https://en.wikipedia.org/wiki/World_Wide_Web),
* **ES1 to Es7 were release over the pass years.**
* **Most modern browser support ES5**
* **Browser support for ES6 still incomplete.**
* **compile to ES5**
* **New features in ES6 like classes,modules,interface,Arrow function**
* **Support for consts**
* **Block scope –let**
* **Async await – 8th edition - 2017**
* **Template literals**
  + **var** customer **= {** name: "Foo" **}**
  + **var** message **=** ‘Hello $**{**customer**.**name**}’;**
* **Promises**
* **De structuring assignment**
* **var** list **= [** 1**,** 2**,** 3 **]**
* **var** **[** a**,** **,** b **] =** list
* **[** b**,** a **] = [** a**,** b **]**
* **var** list **= [** 1**,** 2**,** 3 **];**
* **var** a **=** list**[**0**],** b **=** list**[**2**];**
* **var** tmp **=** a**;** a **=** b**;** b **=** tmp**;**

Architecture:







**Modules:-**

Modules are used in Angular JS to put logical boundaries in your application. Hence, instead of coding everything into one application, you can instead build everything into separate modules to separate the functionality of your application.

Every Angular app has at least one Angular module class, [the *root module*](https://v2.angular.io/docs/ts/latest/guide/appmodule.html),  
conventionally named AppModule.

A module is a mechanism to group components, directives, pipes and services, in such a way that can be combined with other modules to create an application.

The NgModule decorator is used to later on define the imports, declarations, and bootstrapping options.

The BrowserModule is required by default for any web based angular application.

The NgModule decorator requires at least three properties:

* **Export:** This is used to export components, directives, and pipes which can then be used in other modules.

**Imports:** Just like the export array, the import array can be used to import the functionality from other Angular JS modules.

* **Bootstrap:** The bootstrap option tells Angular which Component to bootstrap in the application. The main application view, called the root component, which hosts all other app views. Only the root NgModule should set this bootstrap property.
* **Declaration:** The components, directives, and pipes that belong to this NgModule.
* **Providers:** Creators of services that this module contributes to the global collection of services; they become accessible in all parts of the app.

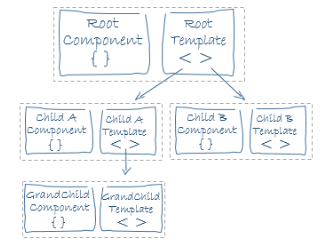
Components:

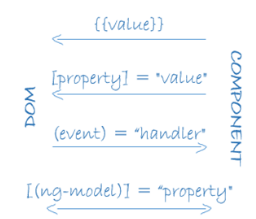
**A component consists of −**

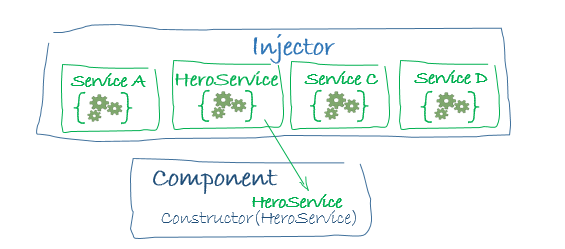
**Class** − this is like a C++ or Java class which consists of properties and methods.

**Metadata** − this is used to decorate the class and extend the functionality of the class.

**Template** − this is used to define the HTML view which is displayed in the application. This part also includes the binding and directives.







Angular can tell which services a component needs by looking at the types of its constructor parameters. For example, the constructor of your HeroListComponent needs a HeroService:

constructor(private service: HeroService) { }

When Angular creates a component, it first asks an **injector** for the services that the component requires.

An injector maintains a container of service instances that it has previously created. If a requested service instance is not in the container, the injector makes one and adds it to the container before returning the service to Angular. When all requested services have been resolved and returned, Angular can call the component's constructor with those services as arguments..

## Lifecycle sequence

**ngOnChanges**: Respond when Angular (re)sets data-bound input properties. The method receives a [Simple Changes](https://angular.io/api/core/SimpleChanges) object of current and previous property values.

Called before ngOnInit () and whenever one or more data-bound input properties change.

**ngOnInIt**: Initialize the directive/component after Angular first displays the data-bound properties and sets the directive/component's input properties. Called once.

**ngDoCheck:** Called during every change detection run, immediately after ngOnChanges() and ngOnInit()

**ngOnDestroy**: Cleanup just before Angular destroys the directive/component. Unsubscribe Observables and detach event handlers to avoid memory leaks.

Event Emitter is a built-in class that ships with Angular that gives us methods for managing and firing custom events.

**Routing**:

Routing helps in directing users to different pages based on the option they choose on the main page. Hence, based on the option they choose, the required Angular Component will be rendered to the user.

Both of the components just render the keywords based on the component. So for the Home component, it will display the Home keyword to the user. And for the About component, it will display the about keyword to the user.

The following points need to be noted about the above program −

* The appRoutes contain 2 routes, one is the Home component and the other is the About component.
* Ensure to declare both of the components.
* The RouterModule.forRoot ensures to add the routes to the application.
* <router-outlet></router-outlet> is the placeholder to render the component based on which option the user chooses.
* Now we have an extra route called path: '\*\*', component: PageNotFoundComponent. Hence, \*\* is for any route which does not fit the default route. They will be directed to the PageNotFoundComponent component.