|  |
| --- |
| Resources |
|  |
|  |
|  |

**Angular**

Angular is a JavaScript binding framework which binds the HTML UI and JavaScript Model.

This helps you to reduce your efforts on writing lengthy lines of codes of binding

It will also help us to build SPA (Single Page Application) by using the concept of routing. It will also have lot of features like HTTP, DI, input, output etc because of which we do not required any other framework.

**Angular v/s Angular JS**

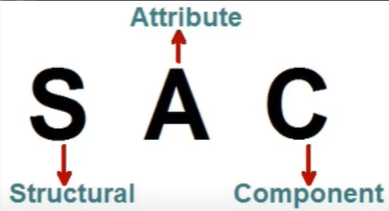
|  |  |  |
| --- | --- | --- |
| Features | Angular JS | Angular |
| Language | JavaScript | TypeScript |
| Architecture | Controller | Component |
| Mobile compliant | No. Not compatible for Mobile apps | Yes. We can build the mobile app using Angular framework like ionic etc. |
| CLI | No.  We don’t have CLI support we need to build app from scratch | Yes. We have awesome CLI support. We can use ng directive from command prompt to create angular project, build, run project and lot of CLI support |
| Lazy loading | No.  we don’t have feature to bundle the components into small pieces. | Yes  We have features to bundle to components into small pieces |
| SEO (Search Engine Optimization) | No | Yes |
| Server Side | No | Yes |

**Directives**

Directives changes the behaviour of HTML DOM

**Types**

1. **Structural Directive** – changes the DOM layout by adding of remove the elements (e.g \*ngFor)
2. **Attribute Directive** – changes the appearance and behaviour of HTML elements (e.g hidden based on true or false it will change the appearance)
3. **Component Directive** – Directives with templates. It is like a user control (writing our own components and rendering templates like custom grids, table etc)



**NPM: Node Package Manager:** Npm is a package manager which makes installation of JavaScript framework easily.

**Node\_modules**: is the folder where all the packages are installed.

**Package.json:** it has all the JavaScript references needed for the project. So rather then installing one package at a time we can install all packages in one go.

What is TypeScript?

* TypeScript is the **Superset** of JavaScript. It added **types** to JavaScript.
* It gives the nice Object-Oriented Programming environment which trans piles/converts to JavaScript.
* So, it is strongly typed we will have less errors because we can do OOP with JavaScript our productivity and quality of code also increases.

**Angular CLI**: Angular CLI is a command line interface by which we can create initial Angular Project template. So rather then starting from scratch we can get some boilerplate code.

**Modules**

* Components where we can write the binding code.
* Modules logically group the components.

**Decorators/Annotation/MetaData**

* Decorator defines what kind of Angular class it is.
* If you decorate “@Component” then it treats as an Angular Component
* If you decorate “@NgModule” ten it treats as an Angular Module

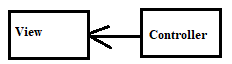
**Teamplates**

* Templates is an HTML view of an Angular in which we can write directives
* There are two ways of defining templates one is **Inline** (using **Template** directive and other is separate HTML file. Which we can refer using **TemplateURL** directive)

**Data Binding & Types**

Data Binding refers to how View and Component is communicating.

1. **Expression/ interpolation** **{{}}** – where data flows from controller to view and mix with HTML markup.

e.g) {{customer.FirstName}} 

1. **Property Binding [] –** where data flows from controller to view and attached to any property

e.g) [disabled] = “customer.FirstName”.

if any changes in components, it will refresh the property.

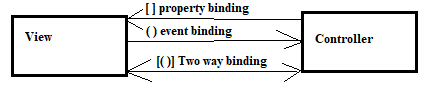
1. **Event binding () –** were data flows from view to components. Something like on click even we are doing some action.

e.g) (click) = “Toggle()”

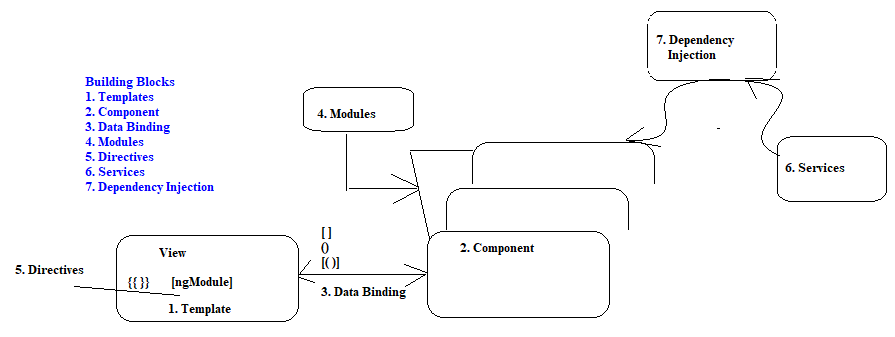


1. **Two-way binding [()]** – data flows from view to component to the vice versa.

**e.g) [(ngModel)] = “Customer.FirstName”**

****

**Angular Architecture/ Building block**

****

1. Templates – the HTML view of an Angular
2. Component – binds view and model
3. Data Binding – defines how views and component communicates
4. Modules – group components logically
5. Directives – changes the HTML DOM Behaviour
6. Services – helps to share common logic across the project
7. DI – dependency injection helps to inject instance across contractor.

**SPA – Single Page Application**

The application where the main UI get loaded only once and the needed UI get loaded on demand.

e.g) Header, Footer & Panel will get loaded once. Based on panel menu item, the area of UI gets refreshed and loaded with new content.

Will load the selected content as configured in Routing.

**Routing**

Routing is the simple collection of two thing url and the components to load based on url.

To implement routing, we required

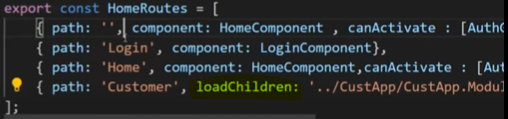
1. Router-outlet
2. Router-link
3. Routing collection

**Lazy Loading**

* It means on demand loading. Loading only the necessary HTML, CSS and JavaScript files so that we can have better performance.

Implementing lazy loading

* First, we need to divide the project into modules
* Second, we need to loadChildren in Module collection. Which indicates to load the relevant module on demand.



**Service**

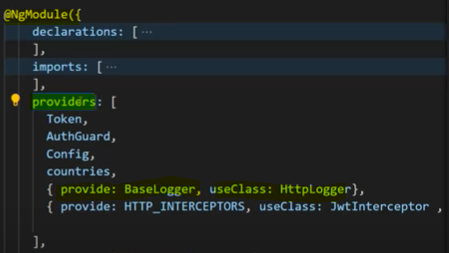
Services help us to share common logic across the angular project

**Dependency Injection**

Dependency injection is an application design pattern were rather than creating object instances from within the components, Angular injects it’s via the constructor

**Implementation**

To implement DI in angular we need to register the class under Providers of @NgModule



**Benefits**

* Decoupling classes – change in one place not in all the places of application

**Ng serve v/s ng build**

* Ng serve builds In Memory while ng build builds on the hard disk
* So when we want to publish the code for PROD/Test we use ng build command. It will create the **dist** folder and place all the bundles java script code.

--prod : ng build –prod flat compress your JS file. Removes comments, creates GUID for you JS files and make application ready for prod.