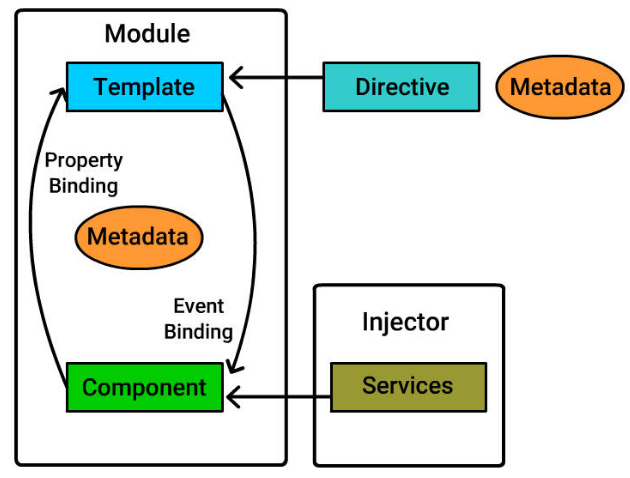
**Angular** – A framework to build client applications in HTML, CSS and JS/TS.

**Why Angular** – Jquery Vanilla JS are hard to maintain and structure with complexity and difficult to test.. Even though there are pattern to maintain JS, it Is complex to understand for a beginner in JS.

**Architecture**



**Angular 1 vs Angular 2/4**

Angular 1 is controller and $scope based while Angular 2/4 is based on an architecture of component hierarchy, dependency injections and directives.

Angular 2/4 is mainly written in Typescript Which introduces features like Class-based Object-Oriented Programming, Generics and Static Typing to framework, which makes it more appealing to developers from non-JavaScript background and leading to shorter development time.

Unlike Angular 1, Angular 2/4 is mainly introduced to be compatible with modern browsers.

Angular 2/4 not only introduces new features like lazy loading/Dynamic loading, Asynchronous template compilation, Simpler Routing and Reactive programming support-using RxJS but also improves on legacy features like dependency injection.

|  |  |  |  |
| --- | --- | --- | --- |
| **AngularJS** | **Angular 2** | **Angular 4** | **Angular 5** |
| View Communicates to Controller through $scope | No Controllers.  Component-based approach |  |  |
|  | Easier to test |  |  |
| AngularJS does not support mobile devices | Supports |  |  |
| Only JS | Offers more language choice for development.  (TS, JS, PureScript) |  |  |
|  | Compiler generated code is a bit huge. | Reduced size of the AoT (Ahead-of-time) compiler generated code, which makes it highly swift and smooth |  |
|  | Supports TypeScript 1.8 | TypeScript 2.1 and TypeScript 2.2 hence new features are available. |  |
|  | <div \*ngIf=”condition”>  <h2>Condition true</h2></div>  <div\*ngIf=”!condition”>  <h2>Condition false</h2></div> | Else block can be used along with \*ngif. So, if you write a code in Angular 2 in this form:  <div \*ngIf=”yourCondition; else myFalsyTemplate”>  <h2>Condition true!</h2> </div>  <ng-template #myFalsyTemplate>  <h2>Condition false!</h2>  </ng-template> |  |
|  |  |  | Eliminates useless code from the application with the help of Build Optimizer for more compressed, arranged and neater code. |
|  |  |  | Is equipped with Angular Universal State Transfer API and DOM Support which allows you to share the state of application between client and server. |
|  |  |  | Incremental compilation for authority and control and get applications compiled fast and easy. |
|  |  |  | Decorator supports lambda expressions and not the nomenclatures or naming functions.  provider: [{  provide: ”  my – service ‘, useFactory: () => null}]}) |
|  |  |  |  |

**Module** –

A Module is a way of organizing related Components, Services, Directives, and Pipes with a specific functionality.

@NgModule is used to declare a Class as Module.

There can be several Modules within an app, but it should consist of at least one root module. Root Module, by convention, must be named: AppModule.

Trick to identify Root Module is that it imports "BrowserModule". Other modules of that same application imports "CommonModule".

imports:[...] - define array of modules required to build application here.

declarations:[...] - define components, directives and pipes for this module here.

bootstrap:[...] - define root component of this module here.

providers:[…] - A provider is an instruction to the DI system on how to obtain a value for a dependency. Most of the time, these dependencies are services that you create and provide.

**Component** –

Basic block of Angular which has view part and core logic.

Can there be multiple root modules?