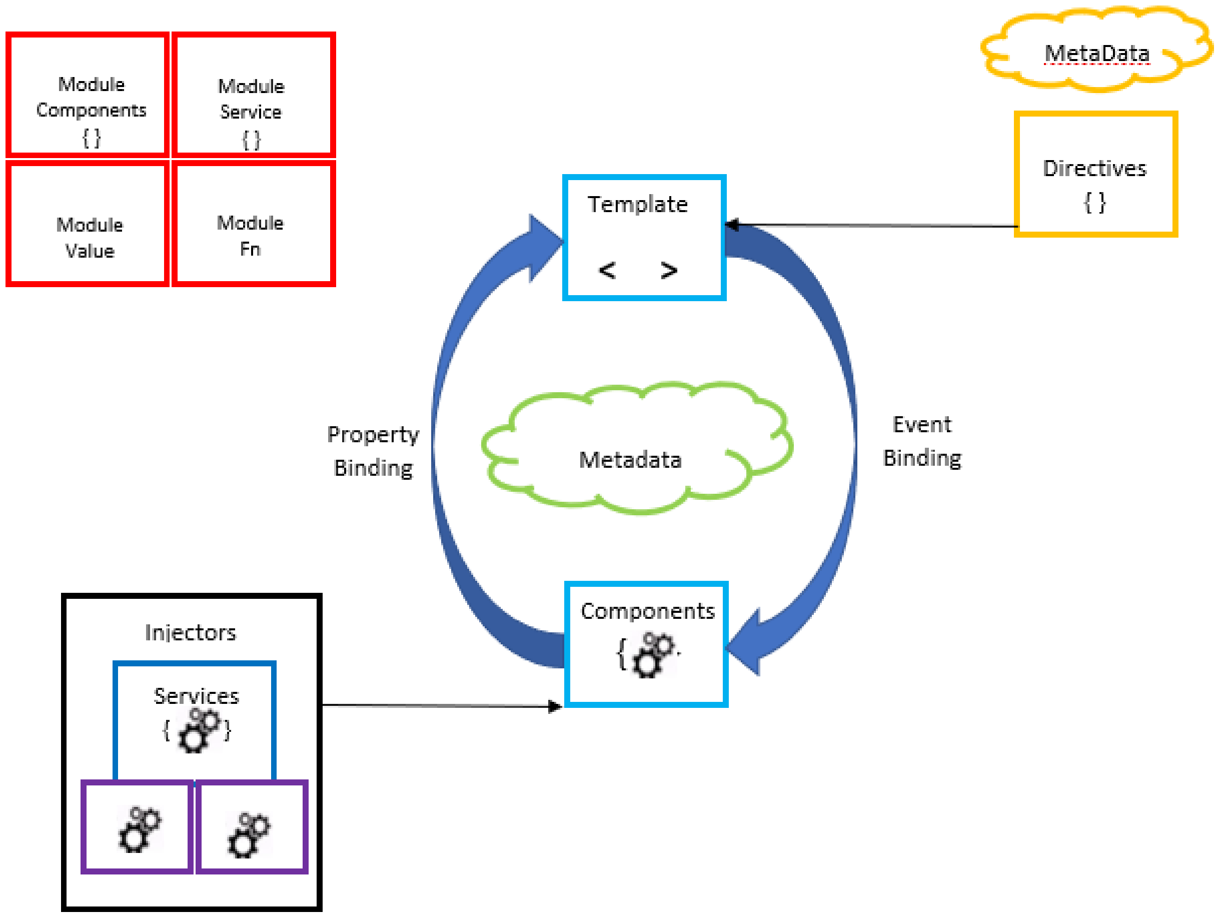
Angular

Angular is one of the most popular frameworks for developing Desktop and mobile applications for clients. Angular application uses HTML and TypeScript. It can also be used in the cross-platform mobile development via IONIC.

Architecture of Angular



The Building blocks of Angular Architecture as depicted in the image are:

1. Template
2. Component
3. Metadata
4. Data Binding
5. Services

6. Directives
7. Dependency Injection

Overview:

1. **Bootstrapping:** This is the process which starts the process of angular application. This process starts in main.ts file which is then compiled into main.js file. It is one of the most essential processes in the working of an application. Bootstrapping only starts the root module and all the other component handling is done by that root module.

2. **NgModule:** It is a manager which handles the compilation part of the application. It works in synergy with other modules. NgModule communicates with other modules for bootstrapping them and works in Parent-Child relationship for the proper execution of the application.

   Here are the properties of NgModule:
   a. **Imports**: When NgModule requires some functionality, which is predefined and present in various modules, it inherits these functionalities using the imports list.

   b. **Exports:** When the NgModule requires to augment functionalities of some other module, this is used.

   c. **Declarations:** When a view is displayed on the screen of any application, it is the mixture of *Component* and *Template* which are contained in declarations. Declaration contains the list of all the *Components* which are going to be present in the particular module. The *Component* contains the data part of the view whereas, *Template* contains HTML tags and Angular markups. When the *Template* requires to share the data with *Component* it uses the process of Event Binding. Whereas, when *Component* requires to send its data to the *Template* it accomplishes it using the Property Binding process.

   d. **Providers**: when there is external work needed to be done in an application like routing data to a server, it is not done using *Components* rather than this *Services* comes into action which is a special concept fabricated to do all these external tasks. Providers contains the list of *Services* present in the module for external work. Providers do not have the actual service but the Blueprint of Services are present.

e. **Injector:** when *Component* requires any particular *service* for example form validation, so rather than requesting to provider, *Component* sends a request to Injector. Injector extracts the blueprint of the Service from the Providers and creates an instance of that service and provides it to the Component. In some cases, the instance of service is already present in an injector, in that case, the service is directly provided to the component, but if there is no instance present than Injector extracts blueprint from the Providers. This process is known as dependency injection.