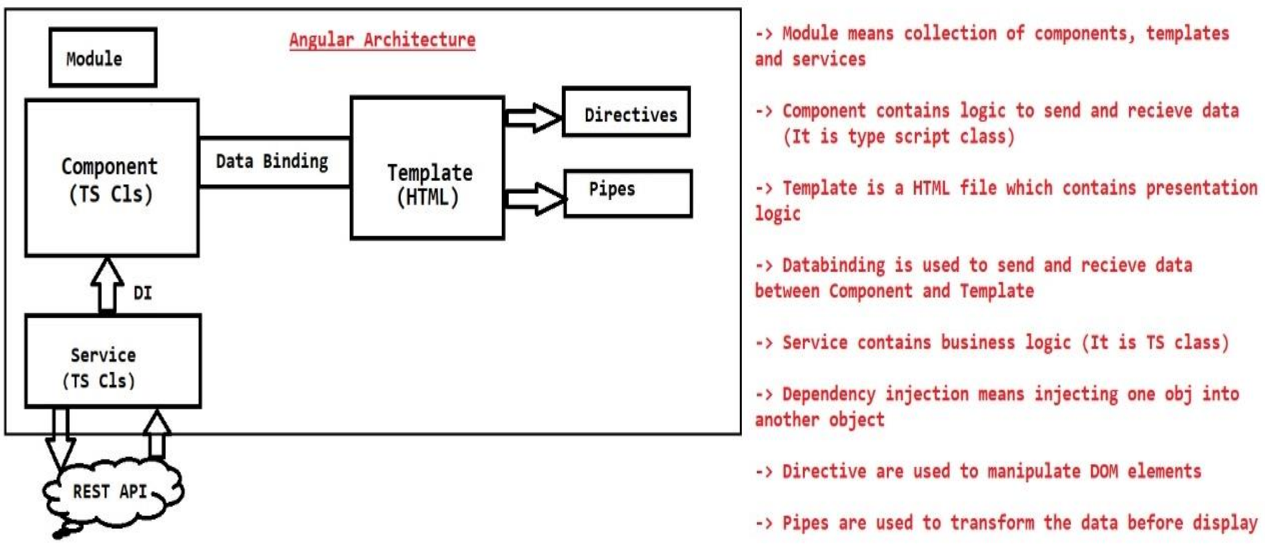
Angular Short Notes



-> Module means collection of components, templates and services

-> Component contains logic to send and recieve data (It is type script class)

-> Template is a HTML file which contains presentation logic

-> Databinding is used to send and recieve data between Component and Template

-> Service contains business logic (It is TS class)

-> Dependency injection means injecting one obj into another object

-> Directive are used to manipulate DOM elements

-> Pipes are used to transform the data before display

- ❖ Angular application is collection of components. In components we will write logic to send data to template and capture data from template. Components are TypeScript classes.
- ❖ Metadata nothing but data about the data. It provides information about components and templates.
- ❖ Template is a view where we will write our presentation logic. In Angular application template is a HTML file. Every Component contains its own Template.
- ❖ Data Binding is the process of binding data between component property and view element in template file.
- ❖ Module is a collection of components, directives and pipes
- ❖ Service means it contains re-usable business logic. Service classes we will inject into Components using Depdency Injection.
- ❖ Dependency Injection is the process of injecting dependent object into target object. In Angular applications services will be injected into components using DI.
- ❖ Directives are used to manipulate DOM elements in the Template.

(We can execute presentation logic based on conditions like if-else , loops etc... using directives)

- ❖ Pipes are used to transform the data before displaying

(lower case to upper case, INR to USD, dd/mm/yyyy to DD-MMM-YYYY)

- In angular application we can have any no.of modules
- When we run angular application, it starts execution from startup module i.e app-module
- Angular application boot strapping will begin from app module
- AppModule will bootstrap AppComponent
- AppComponent is the default component in Angular application
- In Angular application "index.html" file will act as wecome file
- When we access Angular application URL in Browser it will load index.html file
- In index.html file we are using AppComponent selector to invoke AppComponent.

```
<app-root></app-root>
```

❖ Components:

- The component class includes "properties" to store the data, "methods" to manipulate the data.

```
import {Component} from "@angular/core"
@Component (meta-data)
class ClassName{
   property:dataType = value;
   method(args) : returnType {
      //logic
   }
}
```

- selector : represents tag which is used to invoke the component
- templateUrl : represents the html file that has to be rendered when the component is invoked
- template represents content of content
- styleUrls : Represents the list of styles (css) that have to be loaded for the component.
- providers : Represents list of services to be imported into the component
- animations : Represents list of animations to be performed in the component.

Data Bindings

1) Interpolation :
   - It is used to display the value of property in template
   - If the property value is changed then automatically it will be updated in template
   - syntax : {{propertyName}}

2) Property Binding
    - •Property Binding is used to send the data from component to template and assign the same into an attribute of tag.
    - •If the property value is changed then automatically it will be updated in template
      Syntax: [attribute]=*property
3) Event Binding :
    - It is used to pass event notifications from template to component

      Syntax : <tag (event) = "method()" > </tag>

4) Two Way Binding:
    - **"ngModel"** is the pre-defined directive which is used to achieve two-way data binding.
    - Two data binding is applicable **only for <input/> and <select/> tags.**
    - **FormsModule must be imported** in order to use Two Way Data Binding

---

Directives

---

style

- It is used to set CSS property value dynamically at runtime.
- When Component property value changed then CSS property value will be changed automatically.

  Syntax   :   <tagname [style.cssproperty]="component-property">

ngClass

- IT is used to CSS classname dynamically at run time
- Use this directive to set styles with multiple properties conditionally.

Css file :

```
        .class1{

  color:green;

  font-size:30px;
```

html file:

```
 <div [ngClass]="myclass">{{marks}}</div>
```

Ts file :

```
   myclass:string="";

      this.myclass="class1";
```

ngIf

- ➔ The ngIf displays the element if condition is true otherwise it will remove element from DOM.

  `<tag *ngIf="condition"> </tag>`

Note: The ngIf directive must prefix with *

Ts file :

```
    b:boolean;
```

html file :

```
<div *ngIf="b" style="background-color:blue;">Congratulations...!!</div>

<div *ngIf="!b" style="background-color: red;">Better luck next time..!!</div>
```

ngIf and else

- The "ngIf and else" displays one element if it is "true" otherwise it displays other element.

syntax: `<tag *ngIf="condition; then template1;else template2">   </tag>`

```
<ng-template #template1>

...

</ng-template>

<ng-template #template2>

...

</ng-template>
```

ngSwitch

- ➔ The "ngSwitch" checks the value of a variable, weather it matches with any one of the cases and displays element when it matches with anyone.
- ➔ Use "ngSwitch" if you want to display some content for every possible value in a variable.

Syntax:

```
<tag [ngSwitch]="property">

        <tag *ngSwitchCase="'value'"></tag>

        <tag *ngSwitchCase="'value'"></tag>

        <tag *ngSwitchCase="'value'"></tag>
        <tag *ngSwitchDefault></tag>

</tag>
```

ngFor

- ➔ It is used to repeat the tag once for each element in the array. It generates (repeats) the given content once for one element of the array.
- ➔ We have to use use prefix '*' before "ngFor"

Usecase: Displaying all products available in shopping cart.

Syntax: <tag *ngFor="let variable of arrayname">   </tag>

ngFor with Object Array

- ➔ Using this technique we can print array of object values in web page.
- ➔ First we have to store set of objects inside array then read objects one-by-one using "ngFor" and display the data in table format.

Usecase: Reading Product details (name & price) and displaying them.

syntax to create object array

```
arrayRefVariable:classname[] = [
   new ClassName(),  //         new Employee(101, "John", 5000),
   new ClassName(),   //         new Employee(102, "Smith", 5000),
   new ClassName()    //         new Employee(103, "Nick", 6000)
   ];
```

syntax to use object array using ngFor

```
<tag *ngFor="let variable of arrayRefVariable">
     variable.property1
     variable.property2
</tag>
```

ngFor directive with Add and Remove functionality

- ➔ We can allow the user to add new records (objects) to existing array. User can also delete existing records.

Adding element to array

```
arrayVariable.push(value);
```

Removing element from array

```
arrayVariable.splice(index, count);
```