**We are using Angular Version: 5.2.9**  
Q: How to install angular 5?  
A: Angular CLI and its projects have dependencies that needs to have node 6.9.0 or higher along with NPM3 or higher.

\*\*Useful commands while installing angular 5:

**1) How to check npm version installed:**

Ans: npm –v

**2) How to check node version installed**  
Ans: node –v

**3) How to install angular CLI?**  
Ans: npm install -g @angular/cli

This command takes few minutes to complete the execution as it installs angular CLI in your machine.

**4) How to check angular CLI version?**

Ans: ng -v

**5) How to create new folder using command line environment?**  
Ans: **mkdir Projects**

mkdir stands for make directory; this command will create Projects folder on your current destination

**6) How can we change path in Command Line environment?  
Ans: cd Projects**cd stands for change directory, this command will look up for Projects folder that is child of current destination, and all the next operations will execute in the Projects folder.

**7) How can we create new angular 5 Project/application?**

Ans: ng new angularapp

[ angularapp is projectName/application-Name here ], this command takes some minutes to execute, because it creates angular application along with all the necessary files and folders.

**8) How to check content of any folder?**

Ans: use ls command, ls stands for list; it shows content list of that folder.

**9) How to open development/localhost in browser using CLI?**

Ans: Use either ng server --open or ng serve --open

**10) How to stop development/local server?**  
Ans: Press Ctrl + C and then refresh the development server page. You will get the error “The site can’t be reached‘’

**11) How can we create new component through CLI?**  
Ans: ng g c components/navbar [\*\*shorthand command] OR  
ng generate component components/navbar [\*\*LongWay command]  
  
Above command will create component in components folder, if components folder is unavailable then it will create that folder first and then it will create new component in the folder, i.e. **navbar. \*\***We have to click **refresh icon appear on top,** if we unable to see the new component in application.  
  
Every time we run this command, it creates four files, componentName.component.html file, componentName.component.ts file [typescript file], componentName.component.spec.ts file [for testing purpose], componentName.component.css file. And also this command adds this component to @ngModule that is available in **app.module.ts** file.

Note: If you don’t want spec.ts file, then use –-spec=false after the command,

For Ex: ng g c components/navbar –-spec=false

**12) How to create component with inline template and inline style? OR**How can we create component without html and css file?

Ans: ng g c componentName –it –is

-it for inline template and –is for inline style. Above command will add template and styles property to our @component decorator; where we can put our html and CSS code.

**13) How to restart your local/development server if you need it? OR**  
 **In case we make any change in angular.cli json file or if we create any service and register it with appModule what thing we need to do?**  
  
Ans: Use ng serve or npm start command and refresh your DEV server in browser; if it is already running in the browser.

Remember, ng-serve is used to start the DEV server but it does not open your application in the browser. If you want DEV server needs to be start and it opens in the browser then use ng serve --open command.

13) How can we open Visual Studio Code Editor using CLI?  
Ans: code . [code space dot]

**13) How can we include bootstrap CSS into application?**  
Ans: npm install bootstrap@4.0.0-beta.2 jquery popper.js

OR  
 npm install –-save bootstrap

First command will install bootstrap 4.0.0 beta version, and it will also install latest jQuery version and latest popper.js version. Since bootstrap4 has dependency on them we need to install them.  
  
After installing bootstrap, we need to remove caret sign from bootstrap key value available in package.json file. Otherwise it would auto-update your bootstrap version in future.

Then go to angular-cli.json file and give path for below files inside apps array:

1. Give path of bootstrap.css file in styles array
2. Also provide paths for below JS files in scripts array:  
   1) jquery.js file

2) popper.js file

3) bootstrap js file

Note: Every CSS and JS file will be found in **node\_modules** folder.

**14) How to install font-awesome CSS [icon library] in project using NPM?**

npm install font-awesome –-save

After this installation, give this CSS file path into **styles** array available in **apps** array, inside **angular-CLI json file**.

Remember; put this file before bootstrap.css file

**15) How can we create service using CLI?**

A: ng generate service services/data -–module=app.module **OR**

ng g s services/data –-module=app.module

data is service name and within services folder, we are going to create this new service.

This command will create **services** folder **[\*\*if unavailable]** along with data.service.ts file, data.service.spec.ts file and also it will update app.module.ts file.

In command we are writing -–module=app.module, through this we are updating our appModule automatically means we are registering our service with appModule. Within app.module.ts file, it will import the service file and in providers, it will add this service automatically.   
\*\*Remember every service needs to be register in appModule, so

-–module=app.module is always necessary when creating a new service.

We have to click **refresh icon appear on top,** if we unable to see the new service in application.

**16) How can we create module using CLI?**

A: ng g m app-myModule --flat --module=app  
  
It will create one ts file i.e. app-myModule.module.ts and it will also update app.module.ts file

**17) How can we install angular-material using CLI?**

Ans: npm install --save @angular/material

**18) How can we install angular animation?  
A: use below command first:   
npm install @angular/animations@latest –-save  
  
Then go to app.module.ts file, and**

**1) import below module at the top:**

**import { BrowserAnimationsModule } from ‘@angular/platform-browser/animations’;**

**2) Then go to @ngModule and inside @imports array, add  
BrowserAnimationsModule**

**3) Then go to ts file of that component where we are putting animation, at the top import below line:  
import { trigger,style,transition,animate,keyframes,query,stagger } from ‘@angular/animations’;**

4) Then in component.ts file, go to component decorator [@component ]; add animations property and put required animation stuff:

animations: [

trigger('goalsAnimation', [

transition('\* => \*', [

query(':enter', style({ opacity: 0 }), {optional: true}),

query(':enter', stagger('300ms', [

animate('.6s ease-in', keyframes([

style({opacity:0, transform: 'translateY(-75%)', offset: 0}),

style({opacity:.5, transform: 'translateY(35px)', offset: .3}),

style({opacity:1, transform: 'translateY(0)', offset: 1}),

]))

]), {optional: true}),

query(':leave', stagger('300ms', [

animate('.6s ease-in', keyframes([

style({opacity:1, transform: 'translateY(0)', offset: 0}),

style({opacity:.5, transform: 'translateY(35px)', offset: .3}),

style({opacity:0, transform: 'translateY(-75%)', offset: 1}),

]))

]), {optional: true})

])

])

]

'goalsAnimation' is animation name here. In order to use that animation name in our html template; that name needs to add to that element to which we we want to add animation. While adding to html element, The syntax will be [@animationName]=’condition’.   
NOTE: in condition we have written goals.length, it means if any change happens in goals length; like length increase dynamically or decrease dynamically; only in that case animation will work.  
  
**:enter** in query indicates if we are adding element dynamically in the DOM then perform required animation, and **:leave** indicates if we are removing something from the DOM; then perform animation that is mentioned in that respective query.

**------------------------------------------------------------------------------------------------------------------------------------------------------------------------------**

Angular Application:  
**Angular application contains several useful files like: e2e [end to end testing], node\_modules [contains node dependencies], src, package.json, angular-cli.json.**  
  
Application Name:

Src

App

App.component.css

App.component.html

App.component.spec.ts

App.component.ts

App.module.ts

1. **app.component.css:** CSS file that is used to style our application components.
2. **app.component.html:** component’s html file that is used to display our application in the browser. Whenever we run our **DEV server** using **ng-serve --open** command, browser shows us this html file.
3. **app.component.spec.ts:** file that is created for testing purpose.
4. **app.component.ts:** Main component file. It has **@component** section[decorator part], in which we find component-name (selector property), styleUrls array, styles, template/templateUrl etc. And also it contains className section wherein we define our class Properties and methods. And we need to use export keyword in order to import that class in other files.
5. **app.module.ts:** App.module.ts contains @ngModule section. The purpose of @ngModule is to declare everything you create. Like component, service or another module.

@ngModule consists of below sections:

1. declarations,
2. imports &
3. Providers
4. **declarations**: declarations array hold all of the components we create.
5. **Imports**: imports array hold all of the modules we create.
6. **providers**: Providers array hold all of the services we have created.

**And also we have to import all our components & services within app.module.ts file.**

**------------------------------------------------------------------------------------------------------------------------------------------------------------------**

**Property Binding, Class Binding and ngClass, style Binding and ngStyle Examples:**

<input type="text" [id]="myId" value="Vishwas" />

<input type="text" class="{{myClass}}" value="Vishwas" bind-disabled="true" />

<h2 [class.danger]="hasError">Codevolution</h2>

<h2 [ngClass]="multipleClasses">Codevolution</h2>

<h2 [style.color]="'orange'">Style Binding</h2>

<h2 [style.color]="hasError ? 'red' : 'green'">Style Binding</h2>

<h2 [ngStyle]='multiStyle'>using ngStyle</h2>

In above code, multipleClasses & 'multiStyle' both are objects. Through multipleClasses object we can add multiple classes to element & through 'multiStyle' object we can add multiple CSS properties to element.

--------------------------------------------------------------------------------------------------------------------

**Q: What is difference between package.json file and angular.cli.json file?**

**Q: What is difference between app.module.ts and app.component.ts file?**

**Q: What is use of app.component.html file?**

**Q: What is Interface? Why it is useful?**

**Q: What is @ngModule?**

**Q: What happens when we create component using CLI?**

**Q: What is selector property?**

**Q: What will be the value of selector, if we create navbar component?**

**Q: What will be the class name if we create a component navbar?**

**Q: What is ngOnit?**

**Q: What is typescript?**

**Q: What are the types available in typescript?**

**Q: Why we use export and import keywords?**

**Q: What is @component?**

**Q: What are properties and methods in angular?**

**Q: What is \*ngFor?**

**Q: What is \*ngIf?**

**Q: What is ng-template? What is the syntax of ng-template? When it is useful?**

**Q: What are nesting components?**

**Q: What is Pipe Chaining?**

**Using multiple pipes for one element at one go is called as Pipe Chaining.**

**For Ex: suppose we have a property ‘day’  
day = new Date(1992, 3, 1);**

**{{ day | date: ‘fullDate’ | upperCase }}**

**Above we are using date and uppercase pipes with day property.**

**Q: How to create custom pipes?**

**Ans:**

1. **Very first attach your custom pipe to your DOM element; For ex:**

**<li>{{ p.gender | findGender }}</li>**

1. **Then create a ts file that contains custom pipe code; within that file we need to import pipe and pipeTransform from @angular/core**

**import { Pipe, PipeTransform } from '@angular/core';**

**3) Assign pipe name: Then within Pipe decorator, assign pipe name we required.**

**For ex:**

**@Pipe ({ // this is decorator**

**name: ' findGender' // this is pipe Name**

**})**

1. **When we create custom pipe, our class need to PipeTransform as below:**

**export class findGender implements PipeTransform { }**

1. **Now we can insert our logic inside that class now…Remember inside class we, within transform method, we put all the necessary code. And transform method accepts parameter, then process them and finally returns the outcome we need.**

**Suppose we have employee array that contains employee name, age and gender. Genders are in digits format like 1,2 & 3 as follows.**

**myArr2 = [**

**{name: Andrew', age: 32, gender: 1},**

**{name: 'Briyana', age: 27, gender: 2},**

**{name: Brad, age: 28, gender: 1},**

**{name: 'Ronny', age: 26, gender: 1},**

**{name: 'Kim', age: 41, gender: 3},**

**];  
  
  
Our view to display the same is as follows:**

<ul \*ngFor="let p of myArr2">

<li>{{ p.name }}</li>

<li>{{ p.age }}</li>

<li>{{ p.gender | findGender }}</li>

</ul>

**If we want those genders to be convert in apporipriate format like male,female and not disclosed our class code will will be:**

export class findG implements PipeTransform {

transform(value):any {

// value stores array and args stores the string

if(value === 1) {

return "Male";

}

else if( value === 2 ) {

return "FeMale";

}

else if( value === 3 ) {

return "Not Disclosed";

}

}

}

**------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------**

**Q: Suppose we have one array that needs to be sorted in ascending and descending order using custom pipe; in that case our steps will be:**

I) define our array in ts file; for ex our ts file is app.components.ts file  
myArr = [3,6,2,5,4]

II) Create new ts file, this file will contain custom pipe code, let’s say file name is app.sort.ts; we will write code in that file as follows:

import { Pipe, PipeTransform } from '@angular/core';

@Pipe ({ // this is decorator

name: 'sortPipe' // this is our custom pipe Name

})

export class SortPipe implements PipeTransform {

transform(value: any[], args: string):any {

// value stores array and args stores the string

if(args === 'ascending') {

return value.sort();

}

else if( args === 'descending' ) {

return value.sort().reverse();

}

}

}

III) then import our ts file that contains custom pipe code into app.module.ts file. Import ts file first, then put the className into declarations array.iv) Now we can use that custom pipe in our app.component.html file as follows:  
{{ myArr | sortPipe: 'ascending' }}  
this will display our array in ascending order AND  
  
{{ myArr | sortPipe: 'descending' }}  
this will display our array in descending order.

**------------------------------------------------------------------------------------------------------------------------------------------------------------------**

**Q: What are the types of component selectors?**

**A: Three types of component selectors are available:**

**i) Component selector as an element;**

***in component.ts file, write selector as follows:  
selector: ‘app-myComponent’*  
then, in view use this as an html element like**

**<app-myComponent></app-myComponent>**

**ii) Component selector as an attribute  
*in component.ts file, write selector as follows:*  
*selector: ‘[app- myComponent]’*then, in view use this as an html attribute like**

**<div app-myComponent></div>**

**iii) Component selector as an class  
*in component.ts file, write selector as follows:*  
selector: ‘.app- myComponent’  
  
then, in view use this as an CSS class like**

**<div class=”app-myComponent”></div>**

**------------------------------------------------------------------------------------------------------------------------------------------------------**

**Q: How can we create routing-module at the time of project creation?**

**Q: What is data-binding?  
A:**

**Q: What is [innerText] attribute?**

**A: innerText is the property that conditionally displays texts between opening and closing tag.**

**Q: What is [hidden] attribute?**

**Q: What is string interpolation?  
Ans: Expression within double curly braces is called as interpolation. By Interpolation we ask angular to evaluate**

**The content inside the curly braces and display the value when the component is rendered in the browser.  
For Ex. if you have property ‘name’ that hold string value ‘amol’ in that case;**

**If we write {{ name }} then angular will evaluate the value of name property and then in browser we will see ‘amol’ is displaying.**

**Q: What is event binding?  
A: event binding simply means calling a function or executing a code of block on certain event.**

**For ex; on click of button, displaying alert box.   
  
  
Q: What is event binding syntax?**

**Event binding syntax:  
(event) = ‘functionName/code to be executed’;**

**<button (click)=”displayAlert()” ></button>;**

**Q: How can we display spinners in webpage; before displaying the data to user?**

Ans: Use setTimeout and \*ngIf, in \*ngIf directive, use false value first; after particular duration, set the value of that variable to true using setTimeout method

**What is property Binding?**

for ex:

1) <button [disabled]=”!x”></button>

This will disabled our button when if find ‘x’ property is set to false. We are setting disabled attribute dynamically.  
  
2) <img [src]=”user.image”/>  
user object’s image property contains src value of image. src value means path of the image.

**What is class Binding & ngClass?**For Ex:

<li [class.bg-light]=”x”></li>c  
This will add bg-light class to li if x is true.

Here we can check only one condition. The class will be added if condition is true.

ngClass example:  
<span [ngClass]=”hide ? ‘class1’ : ‘class2’”></span>

Here we can check two conditions. If condition is true then add one class and if condition is false then add another class.  
  
<span [ngClass]=”{ ‘hide’: x && y }”></span>

Add hide class if x and y both are true.

**What is styleBinding & ngStyle?**

<li [style.border-color]=”x ? ‘green’ : ‘red’ “></li>

This will add green border-color to li if x is true; and if x false then it will add red border.

**What are Pipes and ngNonBindable? Describe few in-built Pipes?**

**How can we add events in angular?**  
<button (click)=”myFunction()”></button> OR

<button (click)=”fireEvent($event)”></button>  
to pass event.target on click event, We use $event

**Explain few events available in angular?** (click): mouse click event

(dblclick): mouse double click event

(mouseover): mouse hover event  
 (mouseout): leave the element after mouse over on it

(mousedown): similar as mouse-click event  
 (mouseup): When we release the mouse button after clicking an element

(drag): when we drag our element

(keydown)  
 (keyup)

(keypress)  
 (focus)   
 (blur)  
 (copy)  
 (cut)  
 (paste)  
 (submit)

**Q: How can we toggle any element on click event?**

Set property value to false first to hide the element by default, and then on click of element, toggle/reverse its value.  
Ex:   
hide = false; /\* hide property is false here \*/

<button (click) = ”hide = !hide”></button>  
  
<div \*ngIf=”hide”>Content here</div>  
  
Above div will be toggle in order to value available in hide property.  
  
We can use function also in which we can reverse the value of property.

**Q: What is ngModel?**

ngModel is used for two-way data binding. Means changes in the model updates the view and vice-versa.

Syntax:

[(ngModel)]=”property” **Q: What things we need to keep in mind in order to form element work properly?**

If we have form element, we have to follow 2 steps in order to form work properly in angular;

I) In app.module.ts

Import { formsModule } from ‘@angular/Forms’;  
  
II) Then in same page, go to @ngModule, then add formsModule into imports section.

III) Then we need to import ViewChild in component.ts.file. It comes from @angular/core.

**Q: What is the service? OR  
Q: What are the benefits of services?  
Q: How can we implement services in angular?**

**Ans:** Suppose we have two components, **EmployeeName** and **EmployeeDetails**. EmployeeName component displaying employee names; and EmployeeDetails displaying all the Employee Details like employee name, id, age etc. All data comes through **Employee** array that contains same values. Since both components cannot share their data with each other, we will have to create Employee array separately in both components.   
  
But this is not good practice. How can we avoid writing same array twice? To overcome such issues, we use Services. Services are best option that can share the data among multiple components, and also it makes our components lean. Services follows DRY [Don’t Repeat Yourself] policy. Means you need to write code only once and you can reuse it at several places. Services are ideal for AJAX calls using the HttpModule.

**Now, How can we create and implement services in our application?**

1. first create your service using CLI “ng g s command”, let’s say service name is employee.service.ts
2. Then go to service.ts file, in the class, create your custom method; let’s say getEmployees method
3. Now in getEmployees method, return an array that store all the values of Employee Array:

**getEmployees() {  
 return [  
 {“id”:1, “name”:”Andrew”, “age”:30},  
 {“id”:2, “name”:”Brandon”, “age”:25},  
 {“id”:3, “name”:”Christina”, “age”:26},  
 {“id”:4, “name”:”Elena”, “age”:28},**

];  
}

1. **Now Create an empty array in both components;  
   For ex: employees = [];**
2. **Then register your service in AppModule:**

**To register service import your service in AppModule and then put serviceName in Providers array.**

1. **Now in component Constructor, create instance of service  
   For Ex: constructor(private \_employeeService : EmployeeService);  
   \_employeeService is local variable here, that gives us instance of EmployeeService service.**
2. **We have created empty array ‘employees’ [refer step 4], now in ngOnInit method, assign custom service method[getEmployees()] to that empty array i.e.employees  
   this.employees = this.\_employeeService.getEmployees();**

**Through this statement, we are assigning all the array to employees property; because getEmployees() method returns the array that holds all the employee data.  
That’s it. Now we can use Employee Data in both the components.**

**------------------------------------------------------------------------------------------------------------------------------------------------------------------  
  
Que: Steps to implement Angular routing in application?**

**Ans:**1) Firstly when you create your project, put ‘--routing’ at last in your command.  
For Ex. ng new myProject --routing  
above command will create new project along with app-routing.module.ts file. Through that single file, we can handle routing within our project.

2) Set default route: Within routes variable, insert object that hold path and component properties. Set path is blank string and set component to HomeComponent. Setting path equals to blank string means setting the default page which we want to see at first when page done with the loading.

3) Set another routes:

4) Then add routerLink attribute to those elements; on click of them we are redirecting somewhere. When you are putting routerLink don’t forget to remove href attribute if it is there.

**For ex:**<button class="btn btn-dark" routerLink="/posts">Posts Section</button>  
3) set routerLink value for default component as “/” ; empty value in quotes is also allowed.

4) showing active components: add below 2 attributes after routerLink added:  
[routerLinkActive]="['active']"

[routerLinkActiveOptions]="{ exact: true }"  
Both attributes needs to be added to properly display the active elements.

--------------------------------------------------------------------------------------------------------------------------------------------------------------------

**Q: What is WildCard Route?  
Ans: If the user tries to navigate to a route that is not configured, the application will throw an error. For ex. In the URL, if we type routeName that is not configured we will get an error in console. A better way to handle invalid URLs is to use WildCard route. By using Wildcard route, we can navigate the user to ‘404 page not found’ component, if the url is invalid.**

\*\*To create WildCard route, check below solution.

--------------------------------------------------------------------------------------------------------------------------------------------------------------------

**Que: How can we display 404 not found component in routing?  
Ans: If user**1) create a component using “**ng g c componentName**” command; let’s say component name is ‘not-found’

2) Go to not-found.component.html file and put your 404 page not found message there

3) Go to your routing module file (let’s say app-routing.module), and import not-found component there

4) In routes array, insert ‘\*\*’ as a path and Not Found Component Name as your component.

**5) Remember 404 Page not found component route should be at last in routes. It’s because \*\* means anything; and below if we put it at the top then for every URL, 404 page not found component will be displayed.**

**------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------**

What is routing parameter? And how can we retrieve the routing parameter value in project?   
  
Ans: Routing parameter means we are passing parameter that holds some kind of value when we specify routing for our project.

* go to app-routing.module.ts file and add necessary parameter in path in routes array
* specify the parameter value with routerLink: Go to routerLink code; and after route Name specify forward slash along with parameter value
* then go to component.ts file and import ActivatedRoute Module as follows:   
  import { ActivatedRoute } from '@angular/router';
* create instance of ActivatedRoute in constructor as follows: constructor(private route: ActivatedRoute) {}
* and now you can print that value in console using below statement:

this.route.params.subscribe(res => console.log(res.id));  
res for response that we retrieve  
  
--------------------------------------------------------------------------------------------------------------------------------------------------------

**Q: Component Based Router Navigation:**  
Sometimes you may need to change router outlet component based on logic occurring in component class for instance:

* in component.ts file import Router first  
  import { Router } from ‘@angular/router’;
* in constructor, create instance of Router:  
  constructor(private router: Router) {}
* create custom method for ex. sendMeHome  
  sendMeHome() {  
   this.router.navigate([‘’])

}

// Above will redirect to aboutComponent  
  
  
  
OR  
sendMeHome() {  
 this.router.navigate([‘about’])

}

// Above will redirect to aboutComponent

Finally go to component.html page where you want to insert component based router navigation. And on click event call the sendMeHome method or any custom method you may want to call that uses router.navigate method.  
--------------------------------------------------------------------------------------------------------------------------------------------------------------------

**Q: What are template reference variables?**

Ans: Template reference variable is the variable through which we can easily retrieve DOM elements, their attributes, their innerTexts and the values entered in input element etc.

I)Accessing input element’s value:

Check below code, we are accessing value from input on button click; #myInput is template reference variable here.  
**component.html file:**  
<input type="text" #myInput >

<button (click)='showName(myInput.value)'>Click here</button>  
  
**write showName method in component.ts file:**  
showName(value) {

console.log(value);

}  
  
We will see alert displaying the string that we entered in input.  
  
II)Accessing DOM element:   
We can access the DOM element too using template reference variables as follows;  
***component.html file:***  
<input type="text" #myInput>

<button (click)='getInput(myInput)'>Click here</button>  
  
***write getInput method in component.ts file:***  
getInput(value) {

console.log(value);

}  
  
On click of button, we will see the input element in console. It means now we have access of input element.

III) Accessing texts available in the Element:

into html file:   
<p #Para>Hi all</p>  
<button (click)='showTexts(Para.innerText)'>Click here</button>  
  
into ts file:  
 showTexts(value) {

console.log(value);

}  
  
This will display the innerTexts of P element.  
  
IV) Accessing attributes:  
into html file:  
<input type="text" #myInput title="myTitle">

<button (click)='showName(myInput.title)'>Click here</button>  
  
This will display the value of title attribute, i.e. myTitle

--------------------------------------------------------------------------------------------------------------------------------------------------------------------

Q: What is component interaction/ component communication in angular 5?  
Ans: component interaction/component communication means sending data **from** **parentComponent to child component**; and sending data **from ChildComponent to ParentComponent.**Often we need component to be interacted with each other.We use **@input and @output decorators** to achieve this. Through @input decorator child component can receive the data from Parent Component. And through @output decorator, child component can send the data to parent Component.  
  
**A) Receiving data from Parent Component into Child Component:**  
  
Suppose within appComponent we have one child component TestComponent, and appComponent contains one property name = ‘amol’. If we want that property to be used by TestComponent, the steps will be:

1) Define property in appComponent; suppose propertyName is ‘age’

2) in AppComponent html, when we call TestComponent, inside the Testcomponent opening tag, bind that age property using property-Binding.  
For Ex: <app-test [parentData]='age'></app-test>

3) **Now receive the data in testComponent.** Inside ts file of TestComponent, within class , @Input() followed by ParameterName, for Ex @Input()parentData  
  
4) Now import Input at the top of testComponent.

import { Component, OnInit, Input } from '@angular/core';

Now we can use age property inside TestComponent using interpolation like {{ parentData }}  
  
  
**Creating alias for Parent data:**  
If you want to change Parent Component property-Name that we are receciving in childComponent, we need to create alias for that as follows:  
  
@Input('parentData') name;

*This simply means we are creating alias of parentData. Name is the alias for parentData here. {{ name }}***B) Sending data to ParentComponent from Child Component:**  
while sending data to ParentComponent from Child, we need to create custom event using EventEmitter method.  
  
1) First Create instance of EventEmitter class. Through this we create our own custom event.  
childEvent = new EventEmitter();

**childEvent** is the instance of EventEmitter here. It is also called as custom event.  
  
2) Since we use @Output decorator to send the data to ParentComponent, add @Output() decorator to the EventEmitter instance  
@Output() childEvent = new EventEmitter();  
  
3) Import @Output decorator and EventEmitter  
import { Component, OnInit, Input, Output, EventEmitter } from '@angular/core';4) Since without event triggering, we cannot send the data to Parent, we need to fire an event like click, on that event we will send the required data  
For Ex:

**<button (click)='FireEvent()'>Click here to Send Event to Parent</button>**

5) Now write the method which is sending the data using emit class along-with Emitter Instance:  
 **FireEvent() {**

**this.childEvent.emit('Passing Message from Child Component');**

**}**

**6)** In the parent Component html, we need to call our custom event similar like we call angular events in parenthesis.  
For ex: <app-test (childEvent)="message=$event"></app-test>  
Here, $event will hold the value that child component sending to Parent Component. Message is the Parent Component property to which we are assigning value of $event.  
  
7) Finally define the property that holds the value sending by Child Component. In this case, message property needs to be defined in Parent Component. Now we can use child Component value into Parent**. {{ message }}  
8) In above code, we are passing message from Child Component to Parent Component, we can pass property too from ChildComponent. Lets say we have property in childComponent;**

**childProp = 'Hi All';  
Now pass this value when you use emit method.  
this.childEvent.emit(this.childProp);  
emit method is used to send the value from Child Component to Parent Component.**

------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Q: Explain HTTP & Observables?   
A: In angular we use http request & observable to fetch data from the server. For that we need to make http request. The http get request will hit Web-api/web -Service which will fetch the data from DB and send it back as an http response. The response we get back from the http call is called as observable. Often we need to cast/convert observable into an array. A service does this task. And finally we assign that array to a local variable in order to use that received data into application.

We use RxJS external library to work with observables.  
  
**Steps to fetch data from web-server using http and observable:  
1) import HttpClientModule in AppModule**

**import { HttpClientModule } from ‘@angular/common/http’;  
  
Then since it is a module, add it to imports module**

1. **Then go to service file; import HttpClient at the top:  
   import { HttpClient } from ‘@angular/common/http’;**

**Now, within constructor, create instance for HttpClient  
constructor(private http: HttpClient) {}  
  
\*http is a local variable thatis instance of HttpClient**

1. **Now in custom method of service, need to return this.http.get()  
   getEmployees() {  
    return this.http.get();  
   }  
     
   \*\* getEmployees is a custom method here**
2. **Now create a property that holds path of server file that contains data which we are requesting.**

Ex: **private \_url : string = ‘assets/data/employees.json’**;  
  
**employees.json** is an external file that has stored the data. **\_url** is a property here that has stored that file path. Above file is not on server, if our file is on server then we need to provide that path here as a value.

**6)** Now pass that \_url property as an argument to http.get method in custom method :  
 Ex:  **return this.http.get(this.\_url);**

**7)** Now create an ts file that contains interface for the array. Suppose our file name is employee.ts file.  
ex:   
export interface IEmployee {  
 id: number,  
 name: string,  
 age: number  
  
}

1. Now we need to import interface:  
   Ex: import { IEmployee } from ‘./Employee
2. Now go to service file, and assign type to get request

Ex: :  **getEmployees() {   
 return this.http.get<IEmployee[]>(this.\_url)  
}**

It means get request type would be IEmployee array. Means get will expect values from DB exactly as per IEmployee interface.

1. Now we need to import Observable:

Ex: import { Observable } from ‘rxjs/Observable’;

1. assign return type to custom method:  
    **getEmployees(): Observable<IEmployee[]> {   
    return this.http.get<IEmployee[]>(this.\_url)  
   }**It simply means getEmployees method will return the value that would be observable and its type would be IEmployee and it will return array.
2. Now go to component.ts file and create an empty array that is going to store data from external file:  
   For Ex: employees = [];
3. Now in same file, within constructor create instance for ServiceClass:  
   **constructor(private \_employeeService : EmployeeService);**

**14) Now in ngOnInit method, subscribe to Observable**:

**this.\_employeeService.getEmployees().subscribe( data => this.employees = data);  
  
That’s it. Now data has been stored in employees array. We can display that data using ngFor in our application.**

----------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**Q: Explain HTTP Error Handling?  
Ans:   
  
  
Q: Explain Route parameters?  
  
  
Q: Explain Default Routing?**

**Q: Explain Structural Directive?  
Ans: Structural Directives are the directives that let you add or remove html elements from DOM.**

**3 Common structural directives are:  
 a)ngIf**

**b)ngSwitch**

**c)ngFor**

**ngIf and ngSwitch directives are used to conditionally render html elements whereas ngFor is used to render a list of html elements like li element in html.  
  
ngIf examples:**<div \*ngIf = "visible">shown</div>

<div \*ngIf = "!visible">Hidden</div>

<br/>

**ngIf with else statement [ using ng-template element for else block ]:**

<div \*ngIf = "visible; else elseBlock">shown</div>

<ng-template #elseBlock>Else Block</ng-template>

<br/> **ngIf with then and else block:**

**[ using ng-template element for then and else block ]**

<div \*ngIf = "visible; then thenBlock; else elseBlock2"></div>

<ng-template #thenBlock>Then Block</ng-template>

<ng-template #elseBlock2>Else2 Block</ng-template>

ngSwitch:  
**If there are multiple conditions and out of those conditions, one condition is true and we want to perform operation in order to that condition, in that case ng-switch is good option to use.**  
For Ex:  
into html  
<div [ngSwitch]="color">

<div \*ngSwitchCase="'red'">You picked red color</div>

<div \*ngSwitchCase="'green'">You picked blue color</div>

<div \*ngSwitchCase="'blue'">You picked green color</div>  
 <div \*ngSwitchDefault>Please pick either red, green or blue</div>

</div>  
  
into ts file, in class  
**color: string = 'red';**  
   
Output: **You picked red color**

**Note:**

1. **False cases will not be generated in DOM.**
2. **\*ngSwitchCase value will need to write in two quotes: \*ngSwitchCase="'red'"**
3. **ngSwitch does not have asterisk included.**

\*ngFor:  
\*ngFor is used to render a list of elements. We use \*ngFor for retrieve. We often use \*ngFor to fetch data from an array.  
  
Suppose we have an colors array that contains 4 colors  
colors = [‘red’, ‘green’, ‘blue’, ‘black’]  
If we need to display those array elements into our application, We will have to use \*ngFor. our \*ngFor syntax would be as follows:  
**Syntax:**

\*ngFor = ‘let color of colors’  
here let is ES6 feature that indicates block level scope, color refers to each element in the array during the iteration, and colors is our array.  
  
\*ngFor repeats your element as per your array length. For Ex. if in html we have attached \*ngFor to li, that li element will be generated as per the length of an array. In above case 4 li elements will be generated.  
  
We can access index numbers of elements too:   
\*ngFor = ‘let color of colors; index as i’;  
i is a variable that we use to display index numbers here {{ i }}..index starts with zero so first element will be display its index as zero.

We can check whether the element is first or not:  
\*ngFor = ‘let color of colors; first as f’;  
f is a variable that we use to check whether element is first in the array or not.

{{ f }}. It will return boolean value, either true or false.  
  
  
We can check whether the element is last or not:  
\*ngFor = ‘let color of colors; last as l’;  
l is a variable that we use to check whether element is last in the array or not.  
{{ l }}. It will return boolean value, either true or false.  
  
We can check whether the element is even or odd:  
**<div \*ngFor = "let color of colors; odd as o">{{ o }}: {{ color }}</div>  
<div \*ngFor = "let color of colors; even as e">{{ e }}: {{ color }}</div>**

**First statement will return true if element has odd index otherwise it will return false,   
and Second statement will return true if element has even index otherwise it will return false.**

**--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------  
  
Q: What are Pipes?  
Ans: Pipes allow us to transform data before displaying them in view. OR we can say with Pipes we decide how to display the data to user.  
  
Suppose we have four properties in application as follows:  
pipeProp = "Amol";**

**message = "Welcome to codevolution";**

**person = {**

**'firstName' : "John",**

**'lastName' : 'Doe'**

**}**

**date = new Date();  
  
Now we will be use pipes as follows:  
<h2>Pipes:</h2>**

**<h3>String Pipes:</h3>**

**{{ pipeProp }}<br/>**

**{{ pipeProp | lowercase }}<br/>**

**{{ pipeProp | uppercase }}<br/>**

**{{ message | titlecase }}<br/>**

**{{ message | slice : 3 }}<br/>**

**{{ message | slice : 3 : 7 }}<br/>**

**{{ person | json }}<br/>**

**<h3>Number Pipes:</h3>**

**{{ 5.678 | number : '1.2-3' }}<br/>**

**{{ 5.678 | number: "3.4-5" }}<br/>**

**{{ 5.678 | number: '3.1-2' }}<br/>**

**<h3>Percent Pipe:</h3>**

**<p>{{ 0.25 | percent }}</p>**

**<h3>Currency Pipe:</h3>**

**<p>{{ 0.25 | currency }}</p>**

**<p>{{ 0.25 | currency : 'GBP'}}</p>**

**<p>{{ 0.25 | currency : 'GBP': 'code' }}</p>**

**<p>{{ 0.25 | currency : 'Rs.'}}</p>**

**<h3>Date Pipes:</h3>**

**<p>date pipe with date: {{ date | date }}</p>**

**<p>date pipe with short: {{ date | date : 'short'}}</p>**

**<p>date pipe with shortDate: {{ date | date : 'shortDate'}}</p>**

**<p>date pipe with shortTime: {{ date | date : 'shortTime'}}</p>**

**<p>date pipe with long: {{ date | date : 'long'}}</p>**

**<p>date pipe with longDate: {{ date | date : 'longDate'}}</p>**

**------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------**

**Q: What is Dependency Injection?**