Angular Interview Questions.

1.What is Angular?

Angular is typescript based open-source web application framework, Developed and maintained by google.it integrates features such as declarative templates, dependency injection, end-to-end tooling

2.what is the current version of angular?

As of 2023 the current version is 16 released on June 28

3.what are the features of angular?

Component: These are the building blocks of angular application to control html view.

Modules: An angular module is set of directives, component, services etc. an application is divided into logical pieces and each piece of code is called as "module" which perform a single task.

Directives, Testing, Routing

Templates: This represents the view of angular application

Services: it is used to create components which can be shared across entire application.

Metadata: This can be used to add more data to an angular class.

4.Diffrence between Angular and Angular JS?

AngularJS	**Angular**
it is based on MVC architecture	This is based on Service/Controller
This uses JavaScript to build the application	Introduced typescript to write the application
Based on controllers concepts	Component based UI approach
Not a mobile friendly framework	Developed considering mobile platform
Difficulty in SEO friendly application development	Ease to create SEO friendly application

5.what are the advantages of angular?

- Ability to add custom directives.
- Exceptional community support.
- Facilitates client and server communication
- Strong feature such as animation and event handlers
- Follows the MVC pattern architecture
- Offers support for static template and angular template
- Support for two-way data binding
- Supports dependency injection, RESTful services and validations

6.what are the Disadvantages of angular?

- Sometimes building complex **single page application** is really inconvenient and laggy. Because over a period of if size increases the complexity service, components and structure can be little tricky.
- Sometimes dynamic application lags.
- Learning angular requires a decent amount of time and efforts.

7.What is angular material?

- Angular Material is a set of UI component library.
- It provides all the components which requires to build any modern web application.

8. What is Angular CLI?

- CLI(Command line interface) tool that is used to initialize, develop, scaffold, maintain angular application directly from a command shell.
- **Commands:**
    1. ng new
    2. ng generate class
    3. ng generate component
    4. ng generate directive
    5. ng generate pipe
    6. ng generate module
    7. ng generate enum
    8. ng generate service

9.What is TypeScript?

- TypeScript is a typed superset of JavaScript created by, Microsoft that adds. Optional types, classes, async/await and many other features and, compiles to plain JavaScript.
- Angular built entirely on typescript.
- Can define custom data types and make it strictly typed language.
- Current version of typescript is 5.0.

10.what is Angular Universal?

- Angular universal is a server-side rendering module for angular applications.
- This is a community driven project and available under @angular/platform-server-package.
- Recently Angular Universal is integrated with Angular CLI.
- Recently Angular Universal is integrated with Angular CLI.

11.What is Angular Ivy?

- Angular Ivy is a new rendering engine for angular.
- You can choose to opt in a preview version of ivy from angular version 8.

12.what is Bazel Tool?

- Bazel is a powerful build tool developed and massively used by google and it can keep track of the dependencies between different pacakges and build targets
- In angular8, you can build your CLI application with bazel

NOTE: The Angular framework itself is built with bazel

13.Default port which angular apps run on? How can we change it?

- **Default:** 4200
- We can change the port number by passing the command
- **ng serve -port=<new-port-name>**

14.what are the default testing frameworks in Angular applications?

- The default testing frameworks supported by vanilla Angular project are as below

1. **Jasmin:** for unit testing. Written in spec.ts files
2. **Karma:** for test runner and execution
3. **Protractor:** for end-to-end test scripts. It is in e2e file which is located in angular project structure.

15.Explain Package.Json file?

- With the existence of package.json, it will be easy to manage the dependencies of the project.
- If we are using typescript in the angular project then we can mention the typescript package and version of typescript in package.json.

16.Commands to run test cases?

- **Ng test –** to run unit tests.
- **Ng e2e –** for end-to-end test.

17. How do you upgrade angular apps?

- ng update @angular/cli@angular/core

18.How do you convert typescript into javascript?

- It is called **Transpilling**
- Even though typescript is used for writing code in angular applications but it gets internally transpiled into equivalent javascript.

19.How to find out CLI version?

- Ng –version

20.How do you find lint issues in code?

- **Lint issues** -> some alignment, space and syntax error in code.
- **Ng lint** -> to find out lint issues.

21.How can you get the window height and width in component class?

height = window.height

width= window.width

22.where and how to do unsubscribe the Observables?

- **onNgDestroy() method in the component**
- **Subscription.unsubscribe()**

23.how, do you call a method of a service in Component?

- In the template call the method directly
- <div (onClick)=”serviceName.method()”>

24.How to ignore prefix while generating a component?

- Ng g component cl –prefix=false

25.How to optimise an Angular App?

- Consider lazy loading instead of fully bundled app if the app size is more.
- Make sure that 3rd party library, which is not used, is removed from the application.
- Have all dependencies and dev-dependencies are clearly separated.
- Make sure the application doesn’t have un-necessary import statements.
- Consider AOT (Ahead of time) compilation.

26.What is ngZone? Optimise and angular app?

- Ng zone is one of the angular dependencies which provides a mechanism called zones, for encapsulating and intercepting async activities in the browser (eg.setTimeout, setInterval,promises).

27.what are Angular Modules?

- Modules are logical boundaries in your application
- The application is divided into separate modules to separate functionality of our application
- The NgModule decorator has three options

1. **Imports:** to import other dependant modules.
   BrowserModule: it is required by default for any web based angular application.
2. **Declaration:** to define the components of application.
3. **Export:** if we want access of any particular module across entire application we declare it inside export.

4.Bootstrap: which component to bootstrap in application.

5.Providers: we inject all the services under providers.

28.what is the default Module Bundler in Angular Application?

- Webpack

Other packaging and module bundlers that can be used are

- Gulp
- Babel
- Grunt

29. what is ngModule?

- An ngModule class describes how the application parts fit together.
- Every application has at least one ngModule, the root module that we bootstrap to launch the application.

30.what are the metadata for @NgModule?

- **Imports:** To import other dependant modules
- **Declarations:** To import components of application.
- **Exports:**To declare the component which we want to access across the entire application.
- **Providers:** To inject services.
- **Bootstrap :** To define which component need to be bootstrapped.

31.How to create new module in angular application

- Ng generate module <module-name>

32.What are angular components:

- Components are building block of any angular application
- Components are the most basic UI building block of an Angular app which forms a tree of Angular components
- Unlike directives, components always have a template and only one component can be initiated per an element in template.
  example: there should be only one **selector** for one component in html template.

33.Explain components structure and building blocks?

- **Selector:** define the name of the HTML element in which our component will live
- **Template** or **templateUrl –** it can be inline strink or link an external html file. it allows us to tie logic from our component directly to a view
- **Styles –** the styles array for our specific component. We can also link external CSS by **styleUrls.**
- **Directive:** another component directives we want to use inside our components
- **Providers:** This is the place we are passing the services that we need inside our components

34.Can components be nested?

- Yes – components can be nested and can be arranged any way as per the project requirement
- Example:
  Site Framework - module
  Component1
  Component2

35.Explain Angular @component decorator?

- Using @Component decorator we can define the components constituents like tempolateUrl, styleUrls and selector name.

Example:

```
@Component({
  Selector: 'app-global-search',
  templateUrl: './global-search.component.html',
  styleUrls: ['./global-search.component.css']
  })
```

36.Explain Angular Component Lifecycle?

- Angular application goes through an entire set of processes or has a lifecycle right from its initiation to the end of the application.
- The description of each lifecycle method is as below:
  ngOnChanges: When the value of data bound property changes, then this method is called.
  ngOninit: This is called whenever the initialization of the directive/component after Angular first time loads the data bound properties.
  ngDoCheck: This is for the detection and to act on changes that angular can't or won't detect on its own.
  ngAfterContentInit: This is called in response after angular projects external content into the component's view.
  ngAfterContentChecked: This Is called in response after angular checks the content projected into the component.
  ngAfterViewInit: This is called in response after Angular initializes the component's views and child views.
  ngAfterViewChecked: This is called in response after angular checks the component's view and child views.
  ngOnDestroy: This is the cleanup phase just before angular destroys the directive/component

37.What is the difference between constructor and ngOninit?

- **Constructor:** constructor is used to initialize the class. It doesn't have any connection with HTML DOM elements.
- **ngOninit ():** Used to write business logic. Using **ngOninit ().** We can perform actions with the HTML DOM elements.

38.How to add condition to ngClass?

- <app-component [ngClass]=" {'vertical-menu': menuService.isVertical}"></ app-component>

  Note: If **isVertical = true** then vertical-menu class will apply

39.What is Rxjs?

- Rxjs is a library for composing asynchronous and callback-based code in a functional, reactive style using observables.
- Many APIs such as HttpClient produce and consume RxJS Observables and also uses operators for processing observables
- For example, you can import observables and operators for using HttpClient as below

  import {Observables, throwError} from 'rxjs'
  import {catchError, retry} from 'rxjs/operators';

40.what is subscribing?

- An observable instance begins publishing values only when someone subscribe to it.
- So, you need to subscribe by calling subscribe () method of the instance, passing an observer that logs the received message to the console
- Let's take an example of creating and subscribe to a simple observable, with an observer that logs the received message to the console.

```
//Creates an observable sequence of 5 integers, starting from 1
const source = range(1,5);

//Create Observer Object
 Const myObserver = {
 Next: x=> console.log('Observer got a  next value' + x),
  Error: err=> console.error('Observer got an error:' + err),
  Complete: () =>  console.log('Observer got a complete notification),};

// Execute with the observer object and prints out each item
source.subscribe(myObserver);
//=> Observer got a next value:1 and so on till 5
```

41.what is an observable?

- An Observables is a unique Object similar to a promise that can help manage async code.
- Observables are not part of javaScript language so we need to rely on a popular Observable library called RxJS.
- The observables are created using new key word.
- Let see the simple example of observable

```
Import {Observable } from 'rxjs'

Const observable = new Observable(observer => {
  setTimeout(() => {
    observer.next('Hello from Observable!');
},2000);
});
```

42.what are observables?

- Observables are declarative which provide support for passing messages between publishers and subscribers in your application.
- They are mainly used for event handling, async programming, and handling multiple values.
- In this case, you define a function for publishing values, but it will not execute until a consumer subscribes to it.
- The subscribed consumer then receives notifications until the function completes, or until they unsubscribe.

43.What is observer?

- Observer is an interface for a consumer of push-based notifications delivered by an observable.
  It has below structure
- **Interface Observer<T> {**
  Closed?:Boolean;
  next:(value:T) => void;
  error(err:any) => void;
  complete: () => void;
  }

- A handler that implements the Observer interface for receiving observable notifications will be passed as a parameter for observable as below,
  myObservable.subscribe(myObserver);

43. what is the difference between promise and observable?

- Below are the differences

Observable	
Declarative: Computation does not start until subscription so they can be run whenever we need the result	Execute immediately on creation
Provide multiple values over time	Provide only one
Subscribe method is used for error handling which makes centralized and predictable error handling	Push error to the child promise
Provides chaining and subscription to handle complex application	Use only. then() clause

44.What are observable creation functions?

- RxJs provides creation function for the process of creating observable from things such as promises, events, timers and Ajax requests.
- **Create an Observable from promise**
- **Create an observable that creates an AJAX request**
- **Create an observable from a counter**
- **Create an observable from an event**

45.What is multicasting?

- Multi casting is the practice of broadcasting to a multiple list of subscribers in a single execution.
- Let's demonstrate the multi-casting feature,
  var source = Rx.Observable.from([1,2,3)];
  var subject = new Rx.Subject();
  var multicasting = source.multicast(subject);

```
//These are, under the hood, 'subject.subscribe({…})';
multicasted.subscribe({
      next: (v) => console.log('observerA: + v)
});
Multicasted.subscribe({
next: (v) => console.log('observerB: + v)
});
```

46.How do you perform error handling in observables?

- You can handle errors by specifying an **error callback** on the observer

  instead of relying on try/catch which are ineffective in asynchronous environment.

- For Example, you can define error callback as bewlow,

```
myObservable.subscribe({
   next(num) { console.log('Next num:' + num)}
   error(err) {console.log('Received an error:' + errr)}
});
```

47.what are the utility functions provided by RxJS?

- The RxJS library also provides below utility functions for creating and working with observables.
- Converting existing code for async operations into observables
- Iterating through the values in a stream
- Mapping values to different types
- Filtering streams
- Composing multiple streams

48.what is angular Unit testing?

- Angular testing is a type of software testing where we test individual components of an application
- In Angular unit testing is performed using jasmine and karma. Jasmine is the testing framework used for writing the test and karma is used to run tests
- We can also use TestBed and async to test asynchronous code.

49.what is Karma and jasmine?

- **Karma**: is a tool of running tests on browser it lets us spawn browsers and run jasmine tests inside of them.
- **Jasmin:** It is testing framework for javascript programming language that support Behaviour driven development (BDD) software development practice.

50.How to install karma using NPM?

- after installing node js, we can install the karma test runner by running

npm install karma –save-dev command

51.How to define a spec in jasmine?

- **Spec** in jasmine represents a test case inside test suite
- We can define spec by calling the global jasmine function **'it'**, which like **describe** takes a string and a function

```
describe('Suite Name", function() {
    It("test spec", function() {
        expect(expression).toEqual(true);
});
});
```

52. Enlist major matchers available in jasmine?

- toBeArray(), toBeArrayOfBooleans(), toBeFalse(), toBeCalculable(), toBeOddNumber(), toBeemptyObject(), any.after(date) are few built in matchers in jasmine
- For complete list of jasmine matchers please visit Github Jasmine matchers list

53. what Is headless browser?

We know that the User Interface or UI of any software is its most integral part. So, when it comes to Headless Browsers (Yep! You heard it right, it is called "headless"), it means a browser without a user interface or "head." So, when the browser is

headless, the GUI is hidden. Therefore, when you use a headless browser to access any website, you can't see anything. However, the program runs in the background. A headless browser is similar to a normal browser that performs functions such as navigating pages, clicking links, downloading content, and many more. But, with a normal browser, you can check each step and proceed with the help of a GUI presentation. At the same time, you will use Command-line or Console Interface to keep track of changes.

54.Importance of headless browsers?

One clear advantage while using headless browsers is that they are faster than your typical browsers, as you can bypass all the time you take to load the CSS. But this is just one advantage.

Other advantages include:

- **Scraping Websites:** You can scrape the HTML of a website without rendering the full browser.
- **Shorter Development Time:** Checking the code changes for websites from the command line saves developers a lot of time and effort.
- **Performance Monitoring with Headless Scripts:** You can monitor the performance of network applications using headless browsers. Many developers automate screen capture of the website image to check the layouts of their website.

55. what is Angular router?

- Angular Router is a mechanism in which navigation happens from one view to the next as users perform application tasks.
- It borrows the concepts or model of browser's application navigation

56.what is the purpose of base href tag?

- The routing application should add element to the index.html as the first child in the tag in order to indicate how to compose navigation URLs.
- If app folder is the application root then you can set the href value as below <base href="/">

57.Is it mandatory to use <base href=""> tag in application?

- Yes, otherwise the links and routing of the application will not work.

58.What are the router imports?

- The angular router which represents a particular component view for a given URL is not part of angular core
- It is available in library named @angular/router to import required router components
- For example, we import them in app module as below
- Import { RouterModule,Routes } from '@angular/router'

59.What is router outlet?

- The RouterOutlet is a directive from the router library and it acts as a placeholder that marks the spot in the template where the router should display the components for that outlet.
- Router outlet is used like a component
  <router-outlet></router-outlet>
  <! -- Routed components go here -- >

60.what are router links?

- The router link is a directive on the anchor tags which gives the router control over those elements
- Since the navigation paths are fixed, you can assign string values to router-link directive as below.

```
<h1>Angular router</h1>
<nav>
  <a routerLink="/todosList">List of todos</a>
  <a routerLink="/completed">Completed todos</a>
</nav>
<router-outlet></router-outlet>
```

61.What are active router links?

- RouterLinkActive is a directive that toggles css classes for active RouterLink bindings based on the current RouterState. i.e, the Router will add CSS classes when this link is active and remove when the link is inactive.
- For example, you can add them to RouterLinks as below

<h1>Angular router</h1>

<nav>

<a routerLink=”/todosList” routerLinkActive=”active”>List oftodos

<a routerLink=”/completed” routerLinkActive=”active”>Completed todos

</nav>

<router-outlet></router-outlet>

62.what is router state?

- Router state is a tree of activated routes
- Every node in this tree know about the “consumed” URL segment, the extracted parameters, and the resolved data
- You can access the current RouterState from anywhere in the application using Router service and the routerState property after injecting it into constructor.

```
@Component({templateUrl:’’template.html”})
class MyComponent{
  constructor(router:Router) {
    const state: RouterState  = router.routerState;
    const root : Activatedroute = state.root;
   const child = root.firstChild;
   const id:Observable<string> = child.params.map(p => p.id);
  }
}
```

63.What are router events?

- During each navigation, the Router emits navigation events through the Router.events property allowing you to track the lifecycle of the route. The **sequence of router events is as below.**
    1. NavigationStart
    2. RouterConfigLoadStart
    3. RouterConfigLoadEnd
    4. RoutesRecognized
    5. GuardsCheckStart

6. ChildActivationStart
7. ActivationStart
8. GuardsCheckEnd
9. ResolveStart
10. ResolveEnd
11. ActivationEnd
12. ChildActivationEnd
13. NavigationEnd
14. NavgationCancle
15. NavigationError
16. Scroll

64.What is activated route?

- ActivatedRoute contains the information about a route associated with a component loaded in an outlet
- It can be also used to traverse the router state tree
- The Activated route will be injected as a router service to access the information.
- In the below example, you can access route path and parameters

```
@Component({..})

class MyComponent {

  constructor(route:ActivatedRoute) {

    const id: Observable<string>  route.params.pipe(map(p => p.id));

    const url: Observable<string> = route.url.pipe(map(segment => segments.join(‘’)));

//router.data includes both ‘data’ and ‘resolve’

 Const user = route.data.pipe(map(d => d.user));

   }

}
```

65.what is the purpose of Wildcard route?

- If the URL doesn't match any predefined routes then it causes the router to throw an error crash the app
- In this case, you can use wildcard route
- A wildcard route has a path consisting of two asterisks to match every URL.
- For Example, you can define **PageNotFoundComponent** for wildcard route as below
- { path: '**', component : **PageNotFoundComponent** }

66.What are directives?

- Directives is an angular functionality which is used to add behaviour to existing elements of DOM in an application.

67.What are the differences between component and Directive?

- In short note, A component (@Component) is a directive-with-a-template
- Some of the major differences are mentioned in a tabular form

Component	Directive
To register a component we use @Component meta-data annotation	To register a directive we use @Directive meta-data annotation
Components are typically used to create UI widgets	Directives is used to add behaviour to an existing DOM element
Component is used to break up the application into smaller components	Directive is used to design re-usable components
Only one component can be present per Dom element	Many directives can be used per DOM element

68.what are the various kinds of directives?

- There are mainly three kinds of directives

    1. **Components:** These are directive with template
    2. **Structural directives:** These directives change the DOM layout by adding and removing DOM elements.
    3. **Attribute directives –** These directives change the appearance or behaviour of an element, component, or another directive.
       Example -> ngClass, ngStyle, ngModel
    4. **Custom directives –** we can create custom directives using **CLI**
       ng generate directive 'directive-name'

69.what is http client?

- Most of the Front-end applications communicate with back-end services over HTTP protocols using either XMLHttpRequest or fetch() API.
- Angular Provides a simplified client HTTP API known as HttpClient which is based on top of XMLHttpRequest interface.
- This client is available from @angular/common/http package

70.What are HttpClient benefits?

The major advantage of HttpClient can be listed as below:

- Contains testability features – basically we can write test cases to mock http requests.
- Provides typed request and response object
- Intercept request and response – process the data accordingly
- Supports Observable APIs
- Supports streamline error handling

71.Explain on How to use HttpClient with an example?

- **Below are the steps need to be followed for the usage of HttpClient**
    1. import HttpClient into root module
    2. Inject HttpClient into the application
    3. Create a component for subscribing service

Step #1

- Import HttpClient into root module

```
import {HttpClientModule} from '@angular/common/http';
@NgModule({
   Imports: [
     BrowserModule,
     //import HttpClientModule after BrowserModule.
     HttpClientModule
    ],
    })
export class AppModule {}
```

Step #2

- Let's create a userProfileService(userprofile.service.ts) as an example. It also defines get method of HttpClient
-

```
Import { injecatable } from '@angular/core';
Import {HttpClient } from '@angular/common/http';

Const userProfileUrl: string = 'assets/data/profile.json';

@Injectable()
export class UserProfileService {
   constructor(private http:HttpClient) { }

getUserProfile() {
return this.http.get(this.userProfileUrl);
   }
}
```

Step #3

- Let's create a component called
  User ProfileComponent(userprofile.component.ts) which inject
  UserProfileService and invokes the service method

```
fetchUserprofile () {
  this.userProfileService.getUserProfile()
 .subscribe((data:User) =>  this.user = {
     Id: data['userId'],
     name: data['firstName'],
    });
 }
```

72. How can you read full response?

- The response body doesn't may not return full response data because sometimes server also return special headers or status code which are important for the application workflow
- In order to get full response, you should use observe option from HttpClient

```
getUserResponse(): Observable<Httpresponse<User>> {
          return this.http.get<User>(
       this.useUrl, {observe: 'response'});
}
```

73.how to perform error handling?

-
```
fetchUser() {
this.userService.getProfile()
.subscribe(
 (data:user) =>  this.userProfile = {…data}, //success path
 error => this.error = error // error path
   );
}
```

74.what are pipes?

- A pipe takes data as input and transform it to desired output.
  Example: date, uppercase, lowercase, titlecase.
- We can create custom pipes also

75.what is a parameterized pipe?

- A pipe can accept any number of optional parameters to fine-tune its output
- The parameterized pipe can be created by declaring the pipe name with a colon (:) and then the parameter value
- **Example :** Birthday is {{ birthday | date:'dd/mm/yyyy'}} '//18/06/1987

76.how, do you use chain pipes?

- You can chain pipes together in potentially useful combinations as per the needs.

Example : Birthday is {{ birthday | date:'fullDate' | uppercase }}

77.what is custom pipe?

- Apart from built in pipes, we can write our own custom pipe with the below key characters
    1. A pipe is a class decorated with pipe metadata @Pipe decorator, which we import from the core angular library for example
        - **(i)** The pipe class implements the pipeTransform interface's transform method that accepts an input value followed by option parameters and returns the transformed value. The structure of pipe transform would be as below
            - The @Pipe decorator allows you to define the pipe name that you will use within template expressions. It must be a valid javascript identifiers.

78.Give an example of a custom pipe?

- We can create custom reusable pipes for the transformation of existing value. For example, let us create a custom pipe for finding file size based on an extension

```
Import {Pipe, PipeTransform } from '@angular/core';
@Pipe({name:'CustomFileSizePipe'})
export class FileSizePipe implements PipeTransform {
transform(size:number,extension: string='MB'):string {
return (size / (1024 *1024)).toFixed(2) + extension;
}
}
```

79.What is the difference between pure and impure pipe?

- A pure pipe is only called when Angular detects a change in the value or the parameter passed to a pipe
  Example : Any changes to primitive input value (String, Number, Boolean, symbol) or a changed object reference (Date, Array, Function, Object)

- An impure pipe is called for every change detection cycle no matter whether the value or parameter changed.
  Example: An impure pipe is called often, as often as every keystroke or mouse move.

80. what is the purpose of ngFor directive?

- We use Angular ngFor directive in the template to display each item in the list.
  For example, here we iterate over list of users
  <li *ngFor=" let user of users">
  {{user}}
  **
**

81.what is the purpose of ngIf directive?

- Sometimes an app needs to display a view or a portion of a view only under specific circumstances.
- The Angular ngIf directive inserts or removes an element based on a truthy/falsy condition.

82.what happens if we use script tag inside template?

- Angular recognizes the value as unsafe and automatically sanitizes it, which removes the <script> tag but keeps safe content such as the text content of the <script> tag
- This way it eliminates the risk of script injection attacks.
- If you still use it then it will be ignored and a warning appears in the browser console
- Let take an example of **innerHtml** Property binding which causes XSS vulnerability
  export class innerHtmlBindingComponent {
  //for example, a user/attacker-controlled value from URL.
  htmlSnippet = 'Template<Script>alert("Owned")</script>Syntax';

83.what is interpolation?

- **Whenever we bind the data using double curly braces to display from class to view is nothing but string interpolation.**

1. Interpolation is a specific syntax that angular converts into property binding
2. It's a convenient alternative to property binding.
3. It is represented by double curly braces({{}})

84.what are template expressions?

In Simple words we can the method to bind the data is also part of template expression

- A template expression produces a value similar to any javaScript.
- Angular executes the expression and assign it to a property of binding target; the target might be an HTML element, a component , or a directive
- In the property binding, a template expression appears in quotes to the right of the = symbol as in [property]="expression".
- In the interpolation syntax, the template expression is surrounded by double curly braces.
- For example, in the below interpolation, the template expression is {{username}}
- <h3>{{username}}, welcome to angular</h3>

85. what are template statements?

- A template statements responds to an event raised by a binding target such as an element , component, or directive
- The template statements appear in quotes to the right of the = symbol Like **(event)="statement"**

Example: <button (**click)="editProfile()"**>Edit Profile</button>

86. how do you categorize data binding types?

- Binding types can be grouped into three categories distinguished by the direction of data flow. They are listed as below.
    - **From the source-to-View**
    - **From view-to-source**
    - **View-to-source-to-view**

Data Direction	Syntax	Type
From the source-to-view **(one-way)**	1.{{expression}} 2.[target]=”expression” 3.bind-target=”expression”	Interpolation, property binding, attribute, class, style
From view-to-source **(one-way)**	1.(target)=”statement” 2.on-target=”statement”	Event
View-to-source-to-view **(Two-way)**	1.[(target)]=”expression” 2.bind-on-target=”expression”	Two-way

87.What is two-way data binding?

- Two way data binding refers to sharing the data between a component class and its template. if you change data in one place, it will automatically reflect at the other end.
  Example : If you change the value of the input box, then it will also update the value of the attached property in a component class

88.What is property binding ?

- It is used to **pass data from the component** class (component.ts) **and setting the value of the given element with that passed data in the user-end** (component.html)

89.what is data binding?

- Data binding in angular is synchronization between component class and the view, when data changes in component class , the view reflects the change, and when data in the view changes, the component class update as well.

90.what is dynamic component in angular?

- Dynamic Components allow us to create and render components at runtime. This is achieved by referring to a container using a template variable and inject newly created components into it as a result of some user interaction.

91.Advantage of using dynamic component?

- Dynamic component can be used to make the way we render our components across the frontend more flexible and dynamic. They can also be used to decrease your bundle size by importing the component just before creating where needed

92. How can we make service as available across entire application?

- **Using @Injectable ({**
  providedIn =root,
  })

93.what is angular metadata?

- Angular framework is written in typescript language
- Typescript language has concepts of “Decorators”. Using decorators we can add the annotations, we can modify or extend the class and it’s class members. Which Is really makes it so powerful because now using metadata using decorators, we can extend the behaviour of a certain class.
- In angular we call it Metadata that is used for extending classes using decorators For example: A component has decorator and has metadata like styleUrls, templateUrls etc

```
@component({
   Selector:’’jaydeep-component-root”,
   templateUrl:’./app-component.html”,
   styleUrls:[‘./app.component.css’]
})
```

94. what is the actual description of Angular1.x and Angular2+ ?

Angular 1 or till Angular1.8	Angular 2+
❖ Angular JS usually referred to as “Angular.js” or Angular 1.x	❖ Angular 2 is a complete rewrite from the same team that built Angular JS.

❖ It aims to simplify both the development and the testing of application by providing a framework for client-side **model-view-controller (MVC) and (MVVM)** architectures, along with components commonly used in rich internet application	❖ It provides more choice for languages. You can use any of the language from ES5, ES6, Typescript or Dart to write angular 2 code ❖ Angular 1.x was not built with mobile support in mind, where Angular 2 is mobile oriented.
❖ It was initially released in oct 2010	❖ It is written entirely in typescript
❖ Angular JS code is written in javaScript	❖ It was released in September 2016

95. what is decorator in angular?

- In angular decorators are functions that allow a service, directive or filter to be modified prior to its usage. Decorators are design pattern that is used to separate a modification or decoration of class without modifying the original source code.

96.what is AuthGuard and it's benefits?

- Auth guard is an angular route guard used to protect the routes from unauthenticated/unauthorized people. It is implemented using canActivate interface which implements a canActivate function that checks whether the current uuser has permission to activate the requested route.

97. What are the two types of forms in angular?

1.Reactive forms:

- These are also called model-driven forms
- They are created programmatically and provide more control over form inputs and validation
- Reactive forms are more suitable for complex and forms and senario's

2.Template driven forms:

- These forms rely on directives in the template to create and manage the form control objects.
- They are easier to use for simple forms and small-scale applications

98. explain the purpose of FormGroup and FormArray in angular forms?

Form Collection Type	Description	Use Case
FormGroup	A collection of FormControls that allows you to group and manage multiple form controls	Handling the state of a form and validating the form as a whole
Form Array	A collection of FormControls or FormGroups that allows you to manage dynamically growing forms	Adding or removing form controls dynamically based on user input or other conditions

99.what is lazy-loading?

- As Angular generates a SPA(Single page Application), all of its component are loaded at the same time. This implies that a large number of un wanted libraries or modules may also be loaded. Lazy loading in angular is the process of loading website component , modules or other assets when they are needed.

100.Advantage of lazy-loading?

Advantage –

The benefits of lazy loading include:

- Reduces initial load time – lazy loading a web page reduces page weight, allowing for a quicker page load time.
- Bandwidth conservation - lazy loading conserves bandwidth by delivering content to users only if it’s requested