***Top Interview Questions and And Answers Of The Angular: By Pradip [map,filter,set,list,string,custom directive]***

 **Question No:- 1 :- Whats the use of Angular ?**

 Angular is UI binding framework which binds the HTML UI and JS model.

 This helps you to reduce your effort on writing those lengthy lines of code for binding.

 adding to it , it also help you to build SPA by using concept of routing.

 Its also has lot of other features like HTTP, DI, Input output because of which you do not need other framework.

**\*Practical Application of Angular:**

1.Customer relationship management system

2.Travel and booking platform

3.E-commerce platform

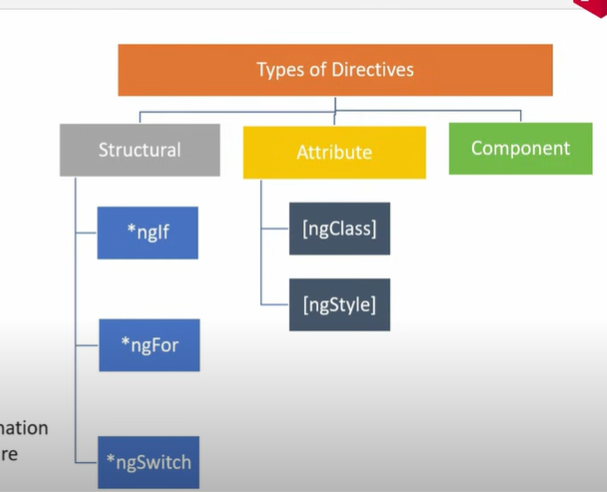
4.Healthcare application

5.Real time chat application

 **Question No:- 2 :- What are directives in Angular ?**

 Directives are markers on HTML DOM file elements that tell angular to attach a behaviour to that element.

 They are a way to extend, transform, and manipulate the DOM and its behavior within Angular applications.

 **Question No:- 3 :- Explain the different types of Angular dir****ectives ?**

Directives can be classified into two main types: structural directives and attribute directives.

 Structural Directives

 Structural directives are responsible for manipulating the structure of the DOM by adding, removing, or replacing elements. They are prefixed with an asterisk (\*) in the template syntax.

 Angular provides built-in directives, such as ngIf, ngFor, and ngSwitch, which are used for conditional rendering, looping through arrays, and switching between multiple views, respectively.

 Additionally, you can also create custom directives to encapsulate and reuse behavior across components.

 Attribute Directives: Attribute directives modify the behavior or appearance of an existing element.

[ngClass]: Dynamically adds or removes CSS classes.

[ngStyle]: Dynamically applies styles.

[(ngModel)]: Two-way data binding for form inputs.

\*component directives: Component directives are used to render content in place of a component.

 **Question No:- 4 :- Explain the importance of NPM and Node\_Modules folder ?**

 Node package manager, which makes installation of java-script framework easily.

 node\_modules is the folder where all the packages are installed.

-NPM (Node Package Manager) is a tool included with Node.js that manages packages (libraries, tools, or frameworks) for JavaScript-based applications, including Angular.

-Manage third-party libraries like RxJS, Angular Material, Bootstrap, etc.

 **Question No:- 5 :- Explain the importance of Package. json file in Angular ?**

1.manage dependency of our project

 It has all the javascript references needed for project. So rather than installing on package at a time we can install all package in one go.

 Whatever JS framework you are using in a project they are all listed in package. json.

 **Question No:- 6 :- What is typescript and why do we need it ?**

 As a name says it adds types to javascript. Typescript is a superset of JS.

 It gives a nice object-oriented programming environment which transpiles / converts to JS.

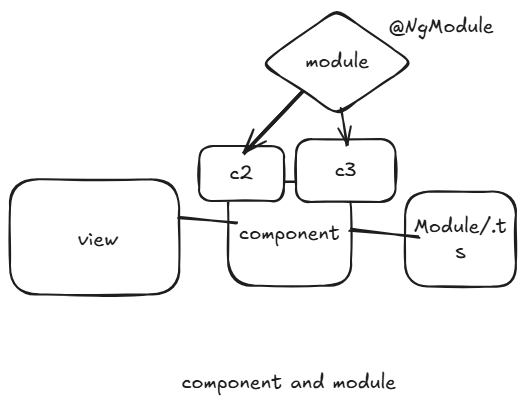
 So as it strongly typed we will have less errors and because we can do OOP with JS our productivity and quality also increases.

 **Question No:- 7 :- Explain importance of Angular CLI ?**

 It stands for Command line interface that simplifies development task

 We can create initial angular project template. It automates tasks such as creating a new project, generating components, building the application, and running a local development server. so rather than starting from the scratch we have some boiler plate code.

 to install CLI the command is - npm install @angular/cli

**Question No:- 8 :- Explain the importance of Component and Modules ?**

Components and modules basic building block of Angular ’s structure/application. Components are like building blocks of the user interface (UI). They define parts of the UI and can be reused throughout the app. Each component has its own template (HTML), styles (CSS), and logic (TypeScript), making the app easier to manage.

Modules help organize the app by grouping related components, services, and other features together. They make the app more scalable and manageable. For example, in a large app, we can have different modules for things like authentication or user management, making the app easier to develop and improving performance.

In short, components help build the UI, and modules keep the app organized and efficient. To make the class as a module we used Ngmodule decorator.

 **Question No:- 9 :- What is a decorator in Angular ?**

Decorators are special types of declarations in TypeScript that are used to modify the structure or behavior of classes or class members. Angular comes with a set of built-in decorators that are used to configure and enhance various elements in an Angular application. Decorators are denoted by the @ symbol followed by the decorator's name.

 Decorators defines what kind of class is it.

 @Component: Used to define a component and its metadata.

 @Directive: Used to define a directive and its metadata.

 @Injectable: Used to define a service and its metadata for dependency injection.

 @NgModule: Used to define a module and its metadata.

 The decorator function receives information about the decorated item and can modify its behavior, add metadata, or perform other tasks.

 The @Component decorator is commonly used to define Angular components. It takes a metadata object as an argument, which provides information about the component.

**Question No:- 10 :- What are Annotation or MetaData (Decorator) ?**

-Annotation is a special label that put in the part of code which tells angular that how should it behave or used.

-It is the way to add the metadata to class or class member using special decorator

-These decorators provide additional information to Angular about how a class or its members should be treated or used.

-To create a component in Angular, you define a TypeScript class and annotate it with the @Component decorator.

**Question No:- 11 :- What is a template ?**

 Template is a HTML view of angular in which we can write directives.

 There are two way of defining template one is inline using template and other is seperate HTML file (provide url of this file in templateUrl)

**Question No:- 12 :- Explain the four types of Data bindings in Angular ?**

 Data binding defines how the view and component communicate with each other. There are four types of the data binding in angular,

 Interpolation binding {{}} or Expression Binding- data flows from the component to the view and we can mix the same with html tags

-String interpolation is only work with string not for number and any other data type.

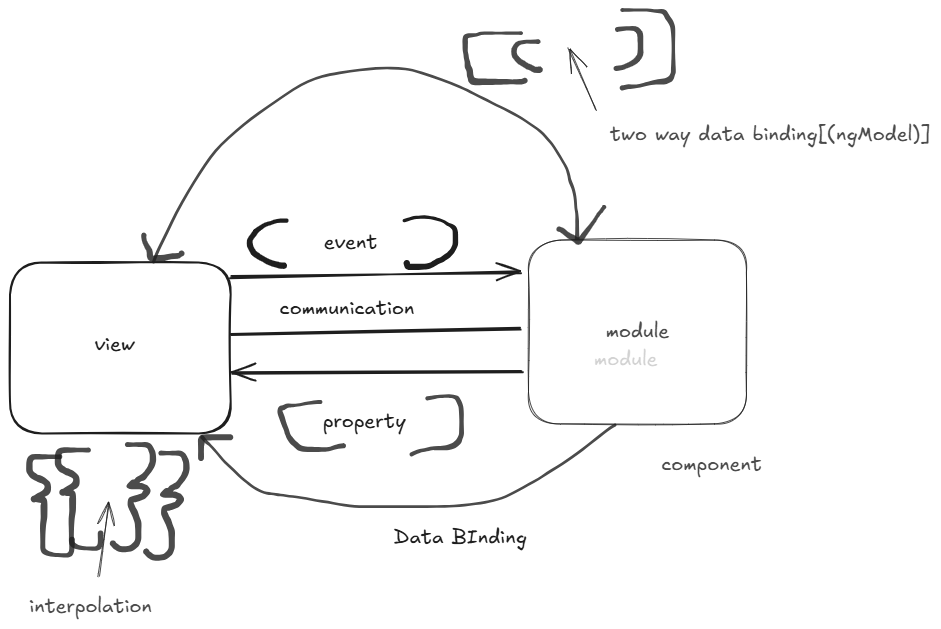
-syntax is simple than property binding.

 property binding [] - data flows from the component to the view

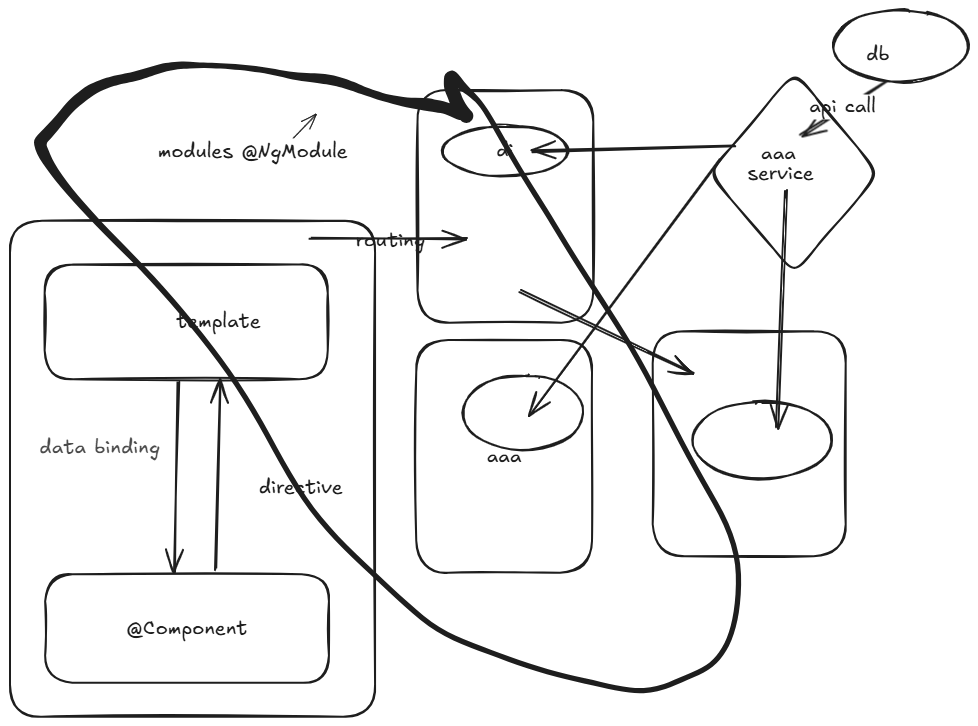
-property binding is superset of string interpolation. It can do whatever interpolation can do. It can set an element property to a non-string data value like boolean.

 event binding () - when you want to send event to the component

 Two way binding [()] - helps you to data flows from the component to the view and vice-versa at the same time syntax:[(ngModel)] //need to import formModule in component.



**Question No:- 13 :- Explain architecture of Angular ?**



1. Template: Html view in angular

2.Component: Binds view and model

3.Modules: Group component logically

4.Binding: Define how view and component communicate

5.Directives: Change Html DOM behaviour

6.Services: Used to share common logic across the project

7.Dependency Injection: help to inject instance across constructor.

8.Routing: used to route from one component to other component

9.Api/Http Call: used to call the data from database /used to retrieve data from database

**Question No:- 14 :- What is SPA in Angular ?**

 SPA stands for single page application

 SPAs are applications where the main UI gets loaded once and then the needed UI is loaded on demand.

 In SPAs, the initial HTML, CSS, and JavaScript resources are loaded when the user first accesses the application.

 Unlike traditional multi-page applications, SPA’s do not require a full page reload with each user interaction. Instead, subsequent interactions or changes in content are handled by dynamically updating the existing page through asynchronous requests.

**Question No:- 15 :- How to implement SPA in Angular ?**

-Using routing we can implement SPA.

-Angular' s router dynamically loads components based on the URL without reloading the entire page.

-Router define navigation

**\*steps to implement spa in angular:**

1.set the collection and define for which url which component is loaded. (app.routing.ts)

const routes: Routes = [

  { path: 'login', component: LoginComponent },

  { path: 'signup', component: SignupComponent },

  { path: 'home', component: HomeComponent },

  { path: “ '', redirectTo: '/login', pathMatch: 'full' }

];

2.after that needs to define <router-outlet><router-outlet/> in html file(app.component.html)

<router-outlet></router-outlet>

3.define the router link in html [routerLink]=”[‘home’]”

-If we want to done routing from .ts file we used this.route.navigate(‘[home]’) -(app.component.ts)

  constructor(private router: Router) { }

 onProfileClick() {

    this.router.navigate(['/profile']);

  }

**Question No:- 16 :- How to implement routing in Angular ?**

 Routing is a simple collection which has two things URL and when this URL called which component to load.

 So routing help you to define the navigation for your angular application

 So if you want to move from one screen to other screen and you want to respect SPA that means not loading and refreshing the whole UI routing is needed.

**Question No:- 17 :- Explain Lazy Loading ?**

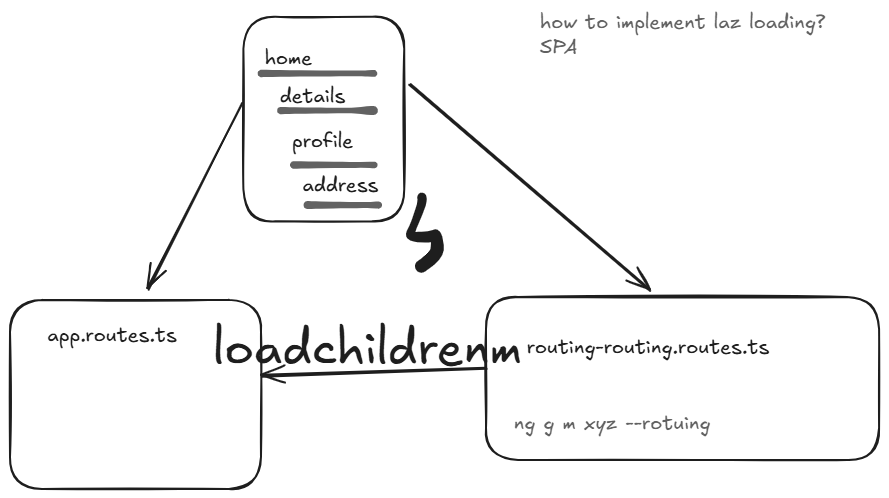
Lazy loading is a technique used in web development to optimize the loading time of a web page by deferring the loading of certain resources until they are actually needed.

In the context of routing in a web application, lazy loading refers to loading specific modules

or components only when the user navigates to a particular route, rather than loading all modules and components at the initial page load.

Lazy loading routing is commonly used in Single Page Applications (SPAs) where the entire application is loaded initially, but certain parts of the application are loaded on-demand as the user interacts with the application.

**Question No:- 18 :- How to implement Lazy Loading in Angular ?**



 Lazy loading involves importing the component only when it's needed. In a lazy-loaded setup, the About Us component would be loaded only when the user navigates to the '/ about-us' route. So improve the application performance by loading only needed component.

 When lazy loading a component, you typically use dynamic imports. Dynamic imports return a Promise, and the component is loaded when the Promise is resolved.

 Note: If you see import() or something similar in your code, it's a good indication that lazy loading is being used.

-For implementing lazy loading we can divide our project into different different module.

-use loadChildren(). //command to create routing file is ng g m xyz --routing

*{*

*path: 'about-us',*

*loadChildren: () => import('./features/about-us/about-us.module').then(m*

*=> m.AboutUsModule)*

*}*

* path: 'about-us':

This specifies the route path. In this case, it's 'about-us'. This means that when the user navigates to the 'about-us' route, the specified module (lazy-loaded) will be loaded and associated with this route.

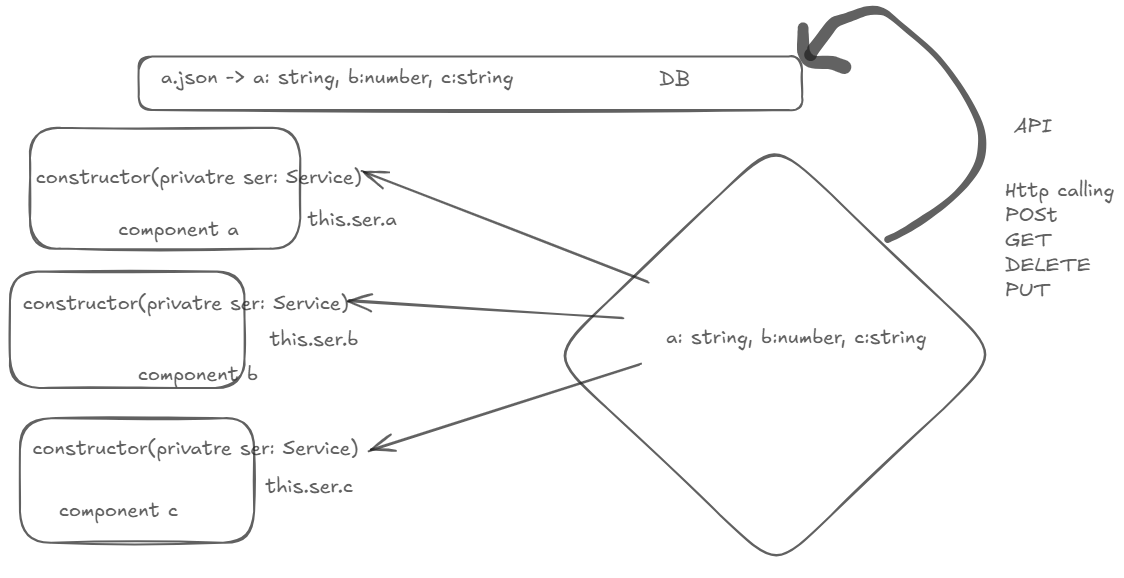
* loadChildren: () => import('./features/about-us/about-us.module').then(m => m.AboutUsModule):

This is the key part that enables lazy loading. Instead of directly importing the module at the time the application loads, it uses the loadChildren property to specify a function that will be called when the module is needed. import('./features/about-us/about-us.module') is a dynamic import statement that returns a Promise. The specified module is not loaded immediately; it will be loaded asynchronously when the Promise is resolved.

 The .then(m => m.AboutUsModule) part ensures that the module is loaded and the specific module class (AboutUsModule) is retrieved. This is essential because Angular expects a dynamically loaded module to have a certain structure, and this structure is defined by the module class.

 In summary, when a user navigates to the 'about-us' route, the associated module (AboutUsModule) will be loaded asynchronously, reducing the initial bundle size and improving the application's loading performance. Lazy loading is especially useful in large applications were loading all modules at once would lead to slower initial load times.

**Question No:- 19 :- Define Services ?**



* Services are used for organizing and sharing code across components.
* Services can be implemented with the help of dependency injection.
* They are singleton objects that can be injected into components, providing a way to encapsulate and share functionality. //ng g s homeservice
* service object get injected in component using constructor
* 90% times services are used to call Api’s .using services we call the API ,distribute data across different component using DI according to their requirement.
* It Allow us to separate business logic from UI logic .we write business logic in service and UI logic in component.
* We can unit test the business logic written in service separetely without creating component .Testing And debugging is easier with service. Also used to share the data between unrelated component
* [Angular Services](https://www.simplilearn.com/tutorials/angular-tutorial/angular-service) perform tasks that are used by multiple components. These tasks could be data and image fetching, network connections, and database management. They perform all the components' operational tasks and avoid rewriting code.

//any event that return the data as observable then we needs to make the event as subscribe

\*Steps to call HTTP request from the service:

1.Inject HttpClient into service by constructor.

2.Make Http call like post, get, put, delete

3.Inject service into corresponding component through the constructor

4.import HttpClientModule into component

5.provides service in Providers array of the component.

6.Fetch data from the service and stored that data into variable

7.use that data into html template by data binding. And performed required operation

(res- json/string/number/list)

@Component module

consturctor(private ser: service) //tells how to provide

imported[ ]

providers[ service] //what to provide

services

**Question No:- 20 :- What is Depedency Injection ?**

 DI is an application design pattern where rather than creating object instances from within the component, Angular injects it via the constructor.

 Dependency injection in Angular allows you to declare the dependencies of a class (like services) in its constructor, and Angular's dependency injection system will provide instances of those dependencies when the class is instantiated.

 Angular's dependency injection system is used to inject services into the components or other services that need them. When a component or another service requests a dependency (like a service) in its constructor, Angular provides the instance of that dependency.

-When we provide service to multiple component ,each component get it own instance of that service.

-Hierarchial injection is possible like service instance(private sub:service ) created in parent component are accessible to its child component also(sub).

**Question No:- 21 :- How to implement Depedency Injection ?**

1.Generate a service using Angular CLI or manually create one consisting of @Injectable decorator.

@Injectable({ [service]

  providedIn: 'root'

})

2.Add the service to the providers array to necessary component whenever required.

@Component({

  selector: 'app-user',

  standalone: true,

  imports: [HttpClientModule],

  providers:[HomeServiceService,SubscriptionService],

  templateUrl: './user.component.html',

  styleUrl: './user.component.scss',

})

3.Use the constructor of the component to inject the service.

  constructor(private userservice: HomeServiceService,private sub:SubscriptionService) {}

4.Angular automatically provides the instance of the service, avoiding the need to create objects manually.

5.Call the methods or properties of the service inside the component to utilize its functionality.

ngOnInit() {

    this.getAllPost()

  }

**Question No:- 23 :- Whats the benefit of Depedency Injection ?**

 This makes the code more modular, testable, and maintainable, as dependencies can be easily swapped or mocked during testing. The @Injectable decorator is an essential part of this mechanism, allowing Angular to understand how to create and manage instances of services.

 DI helps to decouple class dependencies, so that when you add new dependecies do not have change everywhere..

-DI can inherit external logic without knowing how to create it

**Question No:- 24 :- Differentiate between ng serve and ng build ?**

 ng serve build angular applications in memory or in ram, while ng build builds the applications on the hard disk.

-Used for local development.(develop, build, deploy the application in memory)

-No production optimization

 So when you want to go for the production ‘ng build’ command is used.

-Used to prepare the application for deployment.

-Optimized build with --prod flag

**Question No:- 25 :- Explain the --prod parameter in ng build ?**

 Ng build —prod flag compresses your JS file, removes comments, created GUIDs of your JS files and make your application ready for production.

**Question No:- 26 :- Explain ViewChild and ViewChildren?**

 If we want to access DOM elements that are present inside the HTML Template of the same or child component. In that case, we can use View Child and View Children Decorators.

 Also, if we want to access child component properties and methods , that is possible by using View Child and View Children Decorators.

-ViewChild returns the first matching element and ViewChildren returns all the matching elements

**Question No:- 27 :- Why do we need Template reference variables?**

\*Http call in angular Template reference variables give you a way to declare a variable that references a value from an element in your template.

A template reference variable can refer to the following: a DOM element within a template (including custom elements) an Angular component or directive.

1.You can directly access DOM elements in the template using template reference variables.

<input #inputRef type="text" />

<button (click)="logValue(inputRef.value)">Log Value</button>

Here, inputRef is a reference to the input element.

2.They allow you to pass data from a template to a component's method without needing to bind it to a property.

3.You can use template reference variables to interact with child components or directives.

4.Instead of creating complex bindings, template reference variables make it easy to access data or methods directly.

5.They are helpful in dynamic use cases where you need access to multiple elements without creating explicit bindings.

**Question No:- 28:- Differance between angular JS and angular?**

Angular js: language-javascript Angular: Typescript

Architecture: controller Component

mobile compliant: no yes

lazy loading: no yes

server side: no yes

CLI: no yes

design model-mvc component/service

### What is an AOT compilation? What are its advantages?

- AOT(Ahead of Time) ,compiler converts the Angular HTML and TypeScript code into JavaScript during the build phase, i.e., before the browser downloads and runs the code.

This ensures faster rendering to the browser. To specify AOT compilation, include the --aot option with the ng build or ng serve command.

-**Some of its advantages are as follows.**

1. Faster rendering
2. Fewer asynchronous requests
3. Smaller Angular framework download size
4. Quick detection of template errors
5. Better security

**\*What is JIT Compilation?**

**-** The Just-in-Time (JIT) compilation process is a way of compiling computer code to machine code during execution or run time.

-It is also known as dynamic compilation. JIT compilation is the default when you run the ng build, or ng serve CLI commands.

**Question] What is API Calling?**

- Connection of database to angular done by using API calling.,without api angular and database not connected directly.

-We use HttpClient for Api calling inject in our service and needs to import HttpClientModule in component where the service is required.

//(observable-data stream //subscribe -service data disply to component )

-

**<ng-template> is used** to **defer rendering** of its content. It ensures that the content inside it is **not immediately rendered in the DOM** but can be conditionally or programmatically displayed when required.

-Structural directives like \*ngIf, \*ngFor, and \*ngSwitch internally use <ng-template>.

* When such directives are applied, Angular wraps the content in an <ng-template> element, ensuring the DOM elements are not rendered until the directive's condition is satisfied.

**\*Pipes in angular?**

→ In Angular, pipes are a way to transform data in the template before displaying it.

-They are similar to filters in other frameworks and can be used for various tasks, such as formatting dates, converting text to uppercase, or filtering lists.

-Angular provides several built-in pipe and you can also create custom pipes for your specific needs.

Here are a few examples built-in Angular pipes:

**1.  Uppercase and Lowercase Pipes** The uppercase and lowercase pipes transform the input string to uppercase or lowercase, respectively.

**2.  Date Pipe** The date pipe formats a date based on the provided format.

**3.  Percent Pipe** The percent pipe multiplies the input by 100 and appends a percentage sign.

**4.  Json Pipe** The json pipe converts an object to its JSON string representation. It's useful for debugging or displaying structured data

**5.  Keyvalue Pipe** The keyvalue pipe iterates over the key-value pairs of an object, allowing you to display them in a template.

**6.  Slice Pipe** The slice pipe extracts a portion of a string or array. In this example, it extracts characters from index 0 to 6.

**7.  Title Case Pipe** The titlecase pipe converts the input string to title case (capitalizing the first letter each word)

**\*Question:What are pure pipes?**

-**Pure pipes are only executed when the input value changes.**

**-Pure pipes are the default type of pipe in Angular and are optimized for performance.**

**-we must add** pure: true **to the** @Pipe **decorator.**

**\*Question: What is impure pipes?**

-**impure pipes are executed on every change detection cycle**, regardless of whether the input value has changed.

-Impure pipes are useful when the output is dependent on some external state or value is a composite object.

**-we must add**pure: false **to the**@Pipe**decorator.**

**Question: Steps to create custom pipe?**

1.create a typescript class and export it

2.decorate the class with @pipe decorator . Where we specify the name of the pipe.

3.implement PipeTransform interface and ,implement its transform().(value-contain data from component)

4.cmd: ng g p pipename

5.import that pipe in our component and use that like built in pipe

**\*Question: what is pipe transform interface and use?**

->An interface that is implemented by pipes in order to perform a transformation.

Angular invokes the transform () method with the value of a binding as the first argument, and any parameters as the second argument in list form.

**\*Latest Version of Angular?**

→Angular 19.1.2 / 20 jan 2025

-every minor release done in 3 month and major release done by 6 month.

### \*What are Template statements?

**-**Template statements are properties or methods used in HTML to respond to user events.

-For example,

<button (click)="deleteHero()">Delete hero</button>

-The template here is deleteHero(). The method is called when the user clicks on the button.

**\*What is mean by angular universal module?**

→**Angular Universal** is a server-side rendering (SSR) module for Angular applications.

- It allows you to render Angular applications on the server instead of in the browser.

-This is community driven project and available under @angular/paltform-server package

-recently angular universal is integrated with angular cli.

**\*what is the process called by which typescript code get converts into javascript code**

The process of converting TypeScript code into JavaScript code is called **Transpilation**. This is done by the TypeScript compiler, tsc, which reads the TypeScript code and outputs equivalent JavaScript code that can run in any JavaScript environment.

Here’s a quick look at how it works:

1. **Write TypeScript Code**: Developers write their application code in TypeScript.
2. **Transpile**: The TypeScript compiler (tsc) takes the TypeScript code and converts it into JavaScript.
3. **Run JavaScript**: The resulting JavaScript code is then executed by a JavaScript engine, such as the one in a web browser or Node.js.

**\*Change detection?**

-change detection is the mechanism in which angular keeps the template in sync with the component.

**\*Projected content?**

-Html content that present between component selector called project content.

**\*Angular Lifecycle Event?**



A component from creation to destruction goes through several stages and these stages are the life cycle hooks.

Let's go through each one with a brief explanation. Sequence of lifecycle hooks when a component is created:

**Constructor** The constructor of the component class is called first when the class is instatiated. It always run before any angular hook and it is not a part of lifecycle hooks.

**1.**  **ngOnChanges** If the component has input/output bindings, ngOnChanges is called next. This hook is called before ngOnInit and provides information about the changes in the component's input/output propertywhen the binding value changes.

**2.  ngOnInit** s, the ngOnInit lifecycle hook is called. This is called once after the

component is initialized/created. It is ideal for initializing component properties.

After ngOnChange

**3.  ngDoCheck** This hook is called after ngOnInit. It provides an opportunity for the developer to implement custom change detection logic. It is called during every change detection cycle.

**4.ngAfterContentInit** Called after the component's content has been initialized. This hook is often used when you need to perform initialization logic related to content projection

**5.  ngAfterContentChecked** It is called after every check of the component's content. It provides an opportunity to perform logic after the content has been checked during the change detection cycle.

**6.  ngAfterViewInit** Called once after the component's view and child views have been initialized. It is useful for performing operations that require the view to be ready.

**7.  ngAfterViewChecked** It is called after every check of the views of the component. It allows you to perform logic after the views have been checked during the change detection cycle.

**8.  ngOnDestroy** This hook is called just before the directive (or component) is destroyed. It provides an opportunity to perform cleanup logic, such as unsubscribing from observables or releasing resources.

The ngOnDestroy lifecycle hook won't be automatically called for a component unless it is actually destroyed. Components in Angular are typically destroyed when their associated views are removed from the DOM or when their parent component is destroyed, Here are a few scenarios where the ngOnDestroy hook is typically called.

**\*Reactive Form?**

-Reactive forms provide a model-driven approach to handling form inputs whose values change over time.

-Need to import ReactiveFormModule in .ts file

**advantage :**

1.more flexible, but need lot of practice

2.Handle any complex scenario

3.No data binding is done(immutable data model preferred by most developer)

4.More component code less html markup

5.Easier Unit testing

Question No:- 27 :- Why do we need Template reference variables?

* Question No:- 28 :- What is ContentProjection?
* Question No:- 29 :- Explain Content projection Slot?
* Question No:- 30 :- What is ContentChild and ContentChildren?
* Question No:- 31 :- ViewChild vs ViewChildren vs ContentChild vs ContentrChildren?
* Question No:- 32 :- Explain the importance of Component life cycle ?
* Question No:- 33 :- Explain events and sequence of component life cycle ?
* Question No:- 34 :- Constructor vs ngOnInit() ?
* Question No:- 35 :- How to make HTTP calls using Angular ?
* Question No:- 36 :- What is the need of Subscribe function ?
* Question No:- 37 :- How to handle errors when HTTP fails ?
* Question No:- 38 :- How to pass data between components ?
* Question No:- 39 :- Explain importance of input, output & event emitters ?
* Question No:- 40 :- How to pass during routing ?
* **Question No:- 41 :- Is it a good practice to pass data using services ?**

-Yes, using services to pass data in Angular is considered a **good practice** for many scenarios, particularly when you need to share same data across multiple components or manage state globally.

* Question No:- 42:- What is the need of Angular Pipes?
* Question No:- 43:- Can you name some built-in Angular Pipes?
* Question No:- 44:- How to create Custom pipes in Angular?
* Question No:- 45 :- Whats the full form of RxJs?
* Question No:- 46 :- What is the purpose of RxJs?
* Question No:- 47 :- What are observables and observers?
* Question No:- 48 :- Explain the use of Subscribe with sample code.
* Question No:- 49 :- How to unsbscribe in RxJs?
* Question No:- 50 :- Explain concept of operators with sample code.
* Question No:- 51 :- How to install RxJs?
* Question No:- 52 :- Differentiate between promise and RxJs?

**Differance Between Promise and observable?**

both are used to transfer data asynchronously in javascript. otherwise javascript is

single threaded and used to transfer data synchronously.

observable promise

1.Emits multiple values over a period of 1.emits a single value at a time.

time. Also called streaming of data.

2.Are Lazy. They are not executed until 2.Are not lazy. Executed immediately after creation.

we subscribe to them using the

subscribe() method.

3.Have subscription that are cancellable 3.Are not cancellable

using the unsubscribe() method.

* Question No:- 53 :- In Angular where have you used RxJs?
* Question No:- 54 :- Which operators have you used from RxJs?
* Question No:- 55 :- What is Push/reactive vs Pull/Imperative?
* Question No:- 56 :- What are Interceptors in Angular?
* Question No:- 57 :- How to implement Interceptors?
* Question No:- 58 :- Give some use of Interceptors?
* Question No:- 59 :- Can we provide multi-Interceptors?
* 1.Explain Different lifycycle hooks in angular?  
  2.differance between let and const keyword?  
  3.What is Dependency injection in angular?  
  4.do you know about API calling ?  
  5.exaplain Rxjs ? What is observable and observer?
* 6.Different inline element in html?  
  6.is bootstrap is libarary or framework?  
  7.Differance between div tag and span tag?  
  8.differance between padding and margin?
* **Question 60 :- What are two ways of doing validation in Angular?**

-In Angular, there are **two primary ways** to implement form validation:

**1. Template-Driven Validation: This approach is suitable for simple forms and is used with template-driven forms.**

**-Validations are defined directly in the HTML template using Angular directives.**

**-Use Angular’s built-in validation attributes like** required**,** minlength**,** maxlength**, etc.**

**2. Reactive Validation:This approach is suitable for complex and dynamic forms.**

**-Validations are defined programmatically in the component using FormControl and FormGroup classes.**

**-Validations are added using the Validators API.**

* Question 61 :- Template driven forms VS Reactive Forms?
* Question 62 :- In what situations you will use what?
* Question 63 :- Explain template reference variables ?
* Question 64 :- How do we implement Template driven forms?
* **Question 65 :- How to check if overall validation and specific validations are good ?**

**-**In Angular, checking validation involves verifying both the overall form's status and the status of individual form controls. This applies to both **Template-Driven Forms** and **Reactive Forms**.

-Template driven form: Use the valid property of the form.

-Reactive form: Use the valid property of the FormGroup.

* Question 66 :- How do we implement Reactive forms ?
* Question 67 :- How can we implement composite validations?
* Question 68 :- How to create dynamic validation ?
* **Question 69 :- Can you talk about some inbuilt validators ?**

-Angular provides several **inbuilt validators** that are part of the @angular/forms module.

-These validators are used to validate user inputs in both **template-driven** and **reactive forms**.

-They cover common use cases like checking for required fields, minimum/maximum lengths, patterns, and more.

1.Validators.required**: Ensures that the form field is not empty.**

**2.**Validators.minLength**: Validates that the input length is at least the specified number of characters.**

**3.**Validators.maxLength**: Ensures the input does not exceed the specified length.**

**4.**Validators.pattern**: Validates the input against a regular expression (regex).**

**5.**Validators.email**: Ensures that the input is a valid email address.**

* Question 70:- How can you create your own custom validator ?
* **Question 71:- Can we implement angular validators without FORM tag ?**

Yes, Angular validators can be implemented without using a <form> tag. In Angular, the **form structure** is independent of the HTML <form> element. Validators can be applied directly to individual form controls, and you can still validate data without wrapping the controls inside a <form> tag.

* Question 72:- What is [ngModelOptions]="{standalone: true}" ?

Question 73.**What is bootstrap module and bootstrap component?**

-when angular web application start then the first module launched is the bootstrapped module and same is true for bootstrapped component also.

-In Angular, **bootstrapping** refers to the process of initializing and starting the application. It involves setting up the root module and component to launch the app. Here's a breakdown of **bootstrapped module** and **bootstrapped component**:

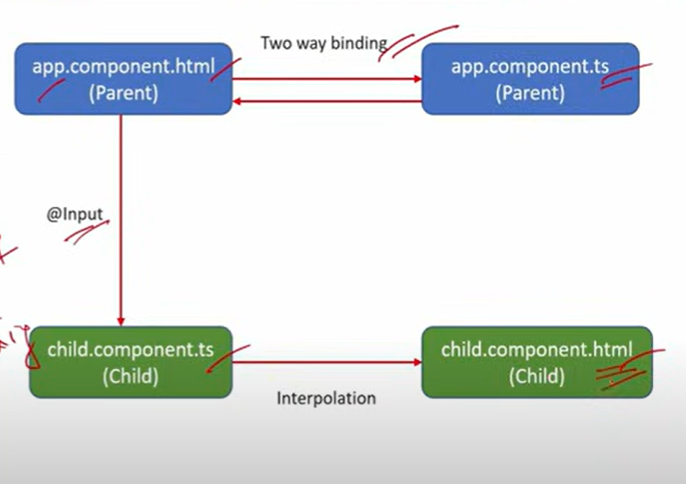
### ****Bootstrapped Module:****

* The **bootstrapped module** is the **root module** of an Angular application, typically called AppModule.
* It is defined in the main.ts file and is the entry point for the Angular app.

### ****Bootstrapped Component****

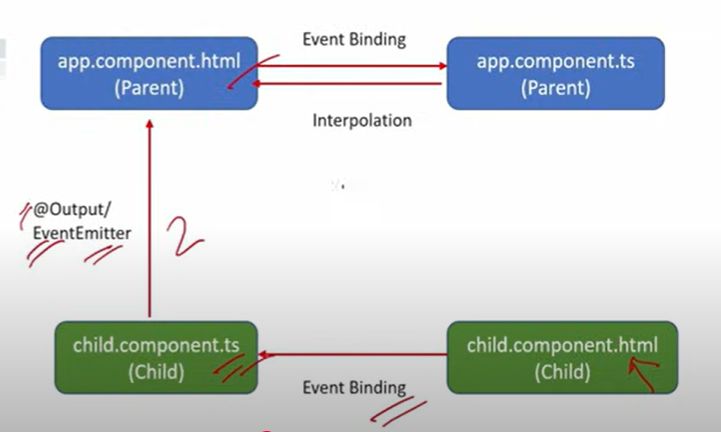
* The **bootstrapped component** is the **root component** of the application.
* It is declared in the bootstrap array of the root module (usually AppModule).
* This component is the starting point for the application's component tree and is responsible for rendering the initial view.

**\*What is @input decorator?How to transfer data from parent component to child component?**



* [-@Input](mailto:-@Input) decorator is used to pass data from parent component to child component.
* Steps:
* -declare the variable in parent
* -then in html view of parent ,within the childselector use property binding for the variable that holds the value from parent.
* -after that declare the value of variable in child class component using @Input() decorator and disply that variable in child component view using string interpolation.

**\*What is @output decorator and event emitter?**



-EventEmitter means discharge event to other component. And @output() means data passed to parent component only.

[-@output](mailto:-@output) decorator and event emitter together are used to pass the data from child component to parent component.

-steps:

-Add a button or any element in the child component's template that triggers the event.

-Use an EventEmitter to emit the data to the parent component.

-In the parent component, listen to the child’s custom event and bind it to a parent function.

-Define the function in the parent component’s TypeScript file to update a variable with the emitted value.

-Use string interpolation to display the variable updated by the emitted data.

\***Reactive Form:**

Reactive Forms offers a model-driven approach that provides greater control and flexibility by managing form controls, validation, and data binding directly within the component class.

* **FormGroup:** Represents a collection of FormControl instances. It is used to group related  form controls, such as those in a form.
* **FormControl:** Represents a single form control. It manages the value and validation status of an individual input field.
* **FormArray:** Manages an array of FormControl or FormGroup instances, allowing for dynamic and repeatable form controls.
* **Validators:** Provides built-in validation functions, such as required, minLength, email,   and min, which can be used to validate form controls.

**Steps:**

* Import ReactiveFormsModule in the AppModule.
* We will create FormGroup and FormControl instances within the component's ngOnInit method, defining form controls and their validations.
* We then bind the form model to the template using [formGroup] and formControlName and handle form submission with a method that processes the form data.

**\*Differance Between Constructor() and ngOnInit():**

**-ngOnInit()**  **Constructor**

1.ngOnInit is an agular lifecycle hook, 1.The constructor is a method in a typescript class ,that

which signals the activation of the automatically gets called when the class is being

created component. Instantiated.

2.ngOnInit is called after 2.constructor is called before any lifecycle hook.

ngOnChanges lifecycle-hook.

3.when ngOnInit called, everything

about 3.when constructor called ,everything in component is component is already ready, so its use is is not ready, so it’s mostly used for injecting

to perform most of the business logic on dependencies only.

component

**\*What is router link in angular?**

-router link is used for navigating to different route or to disply other component.

<a routerLink=”/home”> Home</a>

**\*Role of httpClient in angular?**

-HttpClient is a built -in service class available in angular

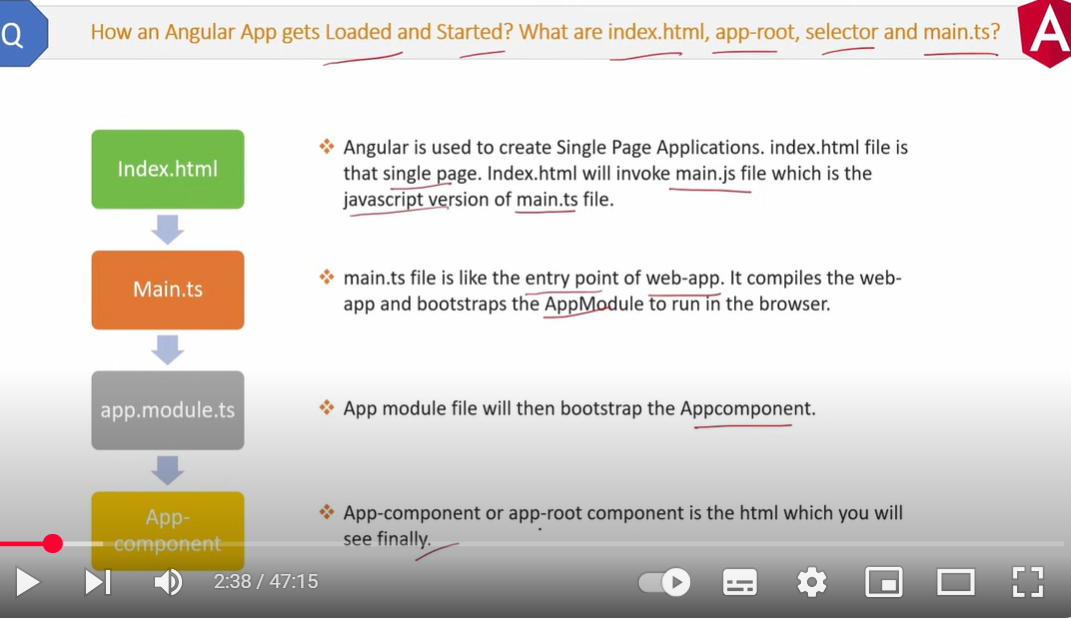
-@angular/common/http package

-perform http request

JSON HTTP

client send a request---------------→http method---------------------------→server sends a response

(GET,POST,PUT,DELETE)



\*Differance Between ReactiveForm And template driven form?

Template driven Form Reactive Form

1.Most code and validation logic is 1.Most code and validation logic is written in component

written in HTML Template. Typescript class file.

2.Have to add FormsModule in 2.Have to add Reactive FormsModule In appModule to

AppModule to activate it. Activate it

3.Used when application is simple and 3.Used when application is complex and have more

have less control controls.

**\*What is Auth Guard or route guard?when to use it?**

* Auth guard is used for guarding the routes. It is used to protect the routes from unauthorized access in angular.
* Command used for creation is ng g guard auth

**How AuthGuard Works?**

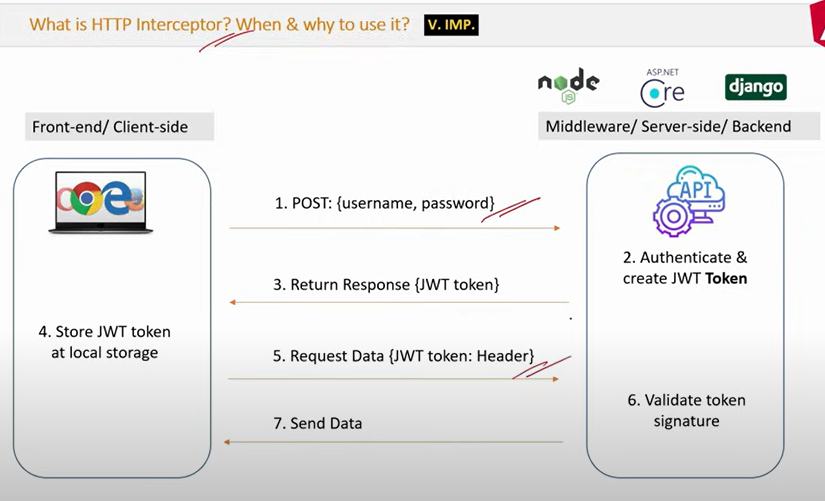
**1.Auth guard provide lifecycle event called canActivate.**

**2.The canActivate is like a constructor. It will be called before accessing the routes.**

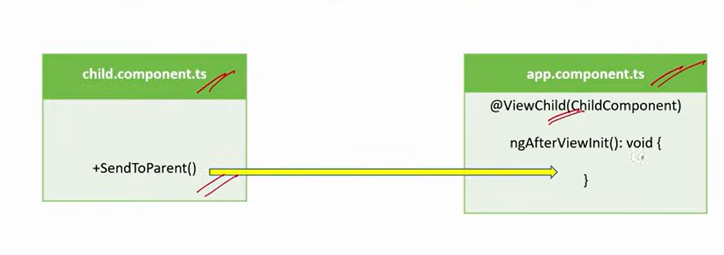
**3.The canActivate has to return true to access the page. If it returns false, we can not access the page.**

**4.We can write our user authorization and authentication logic inside the**

**canActivate function**

**\*What is Http I****nterceptor? When and why to use it?(incomplete)**

**\*How to access the child component from parent component with viewchild.**

-

Steps:

1.Add methos or property in Child component

 sendProperty() {

    console.log('Hi there,I am child component');

  }

2.use child selector in parent html file

<app-child></app-child>

3.in parent(.ts) file use @ViewChild decorator

  @ViewChild(ChildComponent, { static: false }) childComponentProp!: ChildComponent;

//used to get the result from child

  ngAfterViewInit(): void {

    console.log(this.childComponentProp?.sendProperty());

  }