|  |  |
| --- | --- |
| S.no | Questions and Answer. |
| 1. | What is Angular? |
| A. | Angular is a Typescript – based open source web-application framework used to build dynamic single-page applications. |
| 2. | What is Typescript? |
| A. | Typescript is a language that extends Java script by adding static typing, this means that developers can specify types of variables, function parameters, and function return values in their code. This can help catch errors at compile-time instead of runtime, leading to more readable and maintainable code. |
| 3. | What is Component in Angular |
| A | A component in Angular is a building block of an application’s UI that controls a portion of the screen, consisting of an HTML template and a Typescript class.  Here, we import the component decorator from @angular/core. The @Component decorator is used to define the metadata of the component. We define a class called HelloComponent and use the @Component decorator to associate it with selector ‘app-hello’. The selector is used to identify the component when it is used in other templates. The template property defines HTML template for the component. In this case, simply contains an<h1> heading tag with the text “Hello world: |
| 4. | What is a template in Angular? |
| A. | A template is a form of HTML that tells Angular how to render the component. |
| 5. | What is a Module in Angular? |
| A. | A module in Angular is a container for a group of related components, directives, pipes, and services.  For example, in any module it is decorated with the @NgModule decorator, which provides the metadata for the module. The **declarations property** lists the components that belong to this module.  The **imports property** specifies the dependencies of the module.  The **exports property** specifies which components, directives, or other artifacts should be accessible to other modules that import this module. |
| 6. | What is a directive in Angular? |
| A. | A directive in Angular is a class that adds behaviour to an element or component.   1. **<div \*ngIf = “showElement”> This element is conditionally shown</div>** the ngIf directive conditionally adds or removes elements from the DOM based on a given expression. 2. **<ul>  <li \*ngFor=”let item of items”>{{ item }}</li> </ul>** The ngFor directive iterates over a collection and generates HTML elements for each item. 3. **<input type=”text” [(ngModel)]=”username”>** The ngModel directive provides two-way data binding between a form control element and a component property. |
| 7. | What is the difference between a directive and a component in Angular? |
| A. | A directive in Angular is used to modify the behaviour or appearance of an element in the DOM, while a component is a self-contained UI element that defines its own view and behaviour. |
| 8. | What is ngModel directive in Angular? |
| A. | The ngModel directive in Angular is used for two-way data binding, allowing you to bind the value of an input element to a property in the component’s class. |
| 9. | What is ng-content directive in Angular? |
| A. | The ng-content directive in Angular is used to project content from a parent component into a child component’s template, it allows you to create reusable components with customizable content.  **Note**: Needs to practice this. |
| 10. | What is ng-template directive in Angular? |
| A. | The ng-template directive in Angular is used to define a template that can be reused and rendered dynamically in different parts of your application. We use ng-template within ng-container & apply the template using the \*ngTemplateOutlet directive. We reference the template using the local variable #myTemplate defined in the component. The \*ngTemplateOutlet directive instructs Angular to render the template at that location. |
| 11. | What is an ng-container in Angular? |
| A. | The ng-container is a structural directive that provides a way to group elements and apply structural directives like ngIf or ngFor without introducing an additional element into the component.  Use cases of ng-container:   1. Grouping content you can use ng-container to group multiple elements together without introducing an extra element in the DOM. 2. Conditional Rendering ng-container can be used with ngIf to conditionally render a block of elements based on a condition without wrapping them in an additional HTML element. 3. Looping with ngFor ng-container can be used with ngFor to iterate over a collection and render multiple elements without adding an extra element to the DOM.it allows you to apply the top directive to a group of elements. |
| 12. | What is ngClass directive in Angular? |
| A. | The ngClass directive is used to conditionally apply one or more CSS classes to an element based on a specified condition. It allows you to add or remove CSS classes dynamically based on the component’s logic or state. Use cases of ngClass:   1. Object Syntax: in this syntax, the CSS class ‘class-name’ will be added to the <div> element if the condition is true. If the condition is false, the class will be removed. 2. Array Syntax In this syntax, multiple CSS classes are applied to the <div> element. The classes will be added to the element without any condition. 3. String Interpolation 4. Expression Syntax <div [ngClass]=”getClasses()”</div> In the above syntax, a method (getClasses()) is called, which returns an object or an array of CSS classes dynamically based on the components logic. The returned classes will be applied to the element. |
| 13. | What is ngStyle directive in Angular? |
| A. | The ngStyle directive is a built-in directive in Angular that allows you to dynamically set inline styles for HTML elements.  With ngStyle, you can bind an expression to an element’s style property, which enables you to apply styles based on the component’s properties or dynamically calculated values. |
| 14. | What is \* ngFor directive in Angular? |
| A. | The \*ngFor directive is a structural directive in Angular that allows you to iterate over a collection and generate HTML elements for each item in the collection. |
| 15. | What is ngIf directive in Angular? |
| A. | In Angular, **ngIf** is a structural directive used to conditionally include or exclude an element from the DOM (Document Object Model) based on a boolean expression.  <div \*ngIf="isLoggedIn; else loggedOut">  Welcome, User!  </div>  <ng-template #loggedOut>  <div>  Please log in.  </div>  </ng-template> |
| 16. | What is ngSwitch directive in Angular? |
| A. | In Angular, the **ngSwitch** directive is another structural directive that allows you to conditionally render one of multiple elements based on a provided value. It's useful when you have a single expression that can take different values, and you want to render different content based on each value.  <div [ngSwitch]="color">  <div \*ngSwitchCase="'red'">You selected red</div>  <div \*ngSwitchCase="'blue'">You selected blue</div>  <div \*ngSwitchCase="'green'">You selected green</div>  <div \*ngSwitchDefault>Invalid color selection</div>  </div> |
| 17. | What is interpolation in Angular? |
| A. | Interpolation in Angular refers to the process of binding data from a component class to the HTML template. It allows you to dynamically insert values of properties from the component class into the HTML template. |
| 18. | What is data binding in Angular? |
| A. | Data binding in Angular refers to the synchronization of data between the component class and the HTML template.  **Interpolation ({{ }}):** Interpolation allows you to bind data from the component class to the HTML template by placing expressions inside double curly braces. This type of binding is unidirectional, meaning data flows from the component to the template.  Example: {{ name }}  **Property binding ([ ]):** Property binding allows you to bind a property of an HTML element to a property of the component class. It sets the property of the target element to the value of the specified component property.  Example: [src]="imageUrl"  **Event binding (( )):** Event binding allows you to bind an event from the HTML template to a method in the component class. When the specified event occurs, the corresponding method in the component class is executed.  Example: (click)="onClick()"  **Two-way binding ([(ngModel)]):** Two-way binding allows you to bind data in both directions, meaning changes in the UI update the component class and changes in the component class update the UI. It combines property binding and event binding into a single notation.  Example: [(ngModel)]="username" |
| 19. | What is property binding in Angular? |
| A. | Property binding in Angular is a mechanism used to set the value of an HTML element's property to the value of a property in the component class. It allows you to dynamically update HTML element properties based on data or expressions from the component class.  import { Component } from '@angular/core';  @Component({  selector: 'app-my-component',  templateUrl: './my-component.component.html',  })  export class MyComponent {  imageUrl: string = 'https://example.com/image.jpg';  }  Html:  <img [src]="imageUrl" alt="Example Image"> |
| 20. | What is event binding in Angular? |
| A. | Event binding in Angular is a mechanism used to listen for and respond to events raised by HTML elements in the template. It allows you to execute methods in the component class in response to user interactions or other events occurring in the UI.  import { Component } from '@angular/core';  @Component({  selector: 'app-my-component',  templateUrl: './my-component.component.html',  })  export class MyComponent {  handleClick() {  console.log('Button clicked!');  }  }  HTML: <button (click)="handleClick()">Click me</button> |
| 21. | What is two way binding in Angular? |
| A. | Two-way binding in Angular is a powerful feature that allows synchronization of data between the component class and the HTML template in both directions. It combines property binding and event binding into a single notation, enabling automatic updates to the UI when the underlying data changes, and vice versa. Two-way binding is denoted by **[(ngModel)]** in the HTML template.  import { Component } from '@angular/core';  @Component({  selector: 'app-my-component',  templateUrl: './my-component.component.html',  })  export class MyComponent {  username: string = '';  }  **Html:**  <input type="text" [(ngModel)]="username">  <p>Hello, {{ username }}!</p> |
| 22. | What is attribute binding in Angular? |
| A. | Attribute binding in Angular is a way to dynamically set attributes of HTML elements in the template based on values or expressions in the component class. Unlike property binding, which binds to DOM properties and events, attribute binding directly binds to HTML attributes.  import { Component } from '@angular/core';  @Component({  selector: 'app-my-component',  templateUrl: './my-component.component.html',  })  export class MyComponent {  isDisabled: boolean = true;  }  <button [disabled]="isDisabled">Click me</button> |
| 23. | What is class binding in Angular? |
| A. | Class binding in Angular is a mechanism for dynamically adding or removing CSS classes to HTML elements based on certain conditions or expressions from the component class. It allows you to control the styling of elements in the template based on the application state.  import { Component } from '@angular/core';  @Component({  selector: 'app-my-component',  templateUrl: './my-component.component.html',  })  export class MyComponent {  isSpecial: boolean = true;  }  <div [class.special]="isSpecial">Special Content</div>  <div [class.special]="isSpecial" [class.highlight]="isHighlighted">Special Content</div> |
| 24. | What is style binding in Angular? |
| A. | Style binding in Angular is a mechanism for dynamically applying inline CSS styles to HTML elements based on certain conditions or expressions from the component class. It allows you to control the appearance of elements in the template dynamically.  import { Component } from '@angular/core';  @Component({  selector: 'app-my-component',  templateUrl: './my-component.component.html',  })  export class MyComponent {  fontSize: string = '20px';  }  <div [style.font-size]="fontSize">Dynamic Font Size</div> |
| 25. | What is a service in Angular? |
| A. | In Angular, a service is a class that encapsulates reusable functionality and data that can be shared across components, modules, or other services within an Angular application. Services are used to keep components lean and focused on their primary tasks, while business logic, data manipulation, and interaction with external systems are delegated to services.   1. **Data sharing**: Services can manage and share data between multiple components, ensuring data consistency and avoiding code duplication. 2. **Business logic**: Services encapsulate business logic and algorithms, keeping components focused on presentation and user interaction. 3. **HTTP requests**: Services handle HTTP requests and responses, allowing components to communicate with backend servers or external APIs. 4. **State management**: Services can manage application state, providing centralized access to shared state data and ensuring consistency across components. 5. **Dependency injection**: Angular's dependency injection system allows services to be injected into components, making it easy to access their functionality and data. |
| 26. | What is dependency injection in Angular? |
| A. | Dependency injection (DI) in Angular is a design pattern and a built-in mechanism that allows you to inject dependencies (such as services, objects, or values) into components, directives, pipes, or other services. DI helps manage the dependencies of an application by providing a way to create and wire together the various parts of the application in a modular and loosely coupled manner. |
| 27. | What is the @Injectable decorator in Angular? |
| A. | In Angular, the **@Injectable** decorator is used to mark a class as a candidate for dependency injection. When a class is decorated with **@Injectable**, Angular's dependency injection system knows that it may need to inject instances of this class into other components, services, or directives. |
| 28. | What is a Router & Router service in Angular? |
| A. | In Angular, the Router is a built-in library that provides a powerful mechanism for managing navigation between different views or components in a single-page application (SPA). It allows you to define routes for different URLs and map them to specific components, enabling users to navigate through the application by changing the URL in the browser's address bar.  The Router service, provided by Angular's RouterModule, is responsible for performing navigation based on the defined routes. It provides methods and properties for navigating to different routes, accessing route parameters, querying the current route, and more.   1. Some key features: **Route Configuration**: Routes are defined in the Angular application using the RouterModule and its configuration method, **forRoot()** or **forChild()**. Routes consist of a URL path and the corresponding component that should be displayed when the URL matches that path. 2. **RouterOutlet**: The **<router-outlet>** directive is used in the application's template to mark the placeholder where the router will dynamically render the component corresponding to the current route. 3. **Navigation**: The Router service provides methods like **navigate()**, **navigateByUrl()**, and **navigateByData()** to navigate to different routes programmatically. These methods allow you to navigate by specifying the URL or route path, route parameters, query parameters, and other navigation options. 4. **Route Parameters**: Routes can contain parameters that are specified in the URL path, allowing dynamic routing based on user input or data. The Router service provides methods like **getParam()** and **getParamMap()** to access route parameters from the current route. 5. **Route Guards**: Route guards are used to control access to routes and protect sensitive areas of the application. Angular provides various types of route guards, such as CanActivate, CanActivateChild, CanDeactivate, and Resolve, which allow you to implement custom logic for guarding routes. |
| 29. | What is RouterLink directive in Angular? |
| A. | In Angular, the **RouterLink** directive is used to create navigation links in templates that trigger route navigation when clicked. It allows you to navigate to different views or components within the Angular application by defining the target route as an attribute value.  <a routerLink="/home">Home</a>  <a routerLink="/about">About</a> |
| 30. | What is a pipe in Angular? |
| A. | In Angular, a pipe is a feature that allows you to transform data values within a template before displaying them to the user. Pipes are used to format data, apply filters, and perform other transformations in a declarative way, directly within the HTML templates. |
| 31. | What is the async pipe in Angular? |
| A. | 1. In Angular, the **async** pipe is a built-in pipe that facilitates the handling of asynchronous data streams in templates. It subscribes to an Observable or a Promise and then returns the latest value emitted by the Observable or resolved by the Promise. The **async** pipe also manages the subscription and unsubscription process automatically, helping to prevent memory leaks by unsubscribing when the component is destroyed. **Here is how it works:** **Subscribing to Observable or Promise**: When used in a template, the **async** pipe subscribes to an Observable or a Promise provided by the component. 2. **Displaying the latest value**: The **async** pipe displays the latest value emitted by the Observable or resolved by the Promise in the template. 3. **Handling changes**: If the Observable emits a new value or the Promise is resolved with a new value, the **async** pipe automatically updates the displayed value in the template. 4. **Automatic cleanup**: When the component containing the **async** pipe is destroyed (e.g., when navigating away from the component), the **async** pipe automatically unsubscribes from the Observable or Promise, preventing memory leaks. |
| 32. | What is the pure pipe in Angular? |
| A. | Pure pipes are stateless and deterministic. They always produce the same output for a given input and don’t have any internal state. They optimise performance by running only when their input values change. |
| 33. | What is impure pipe in Angular? |
| A. | Impure pipes can have internal state or rely on external factors that can change during the application’s execution. They are called more frequently as they are not optimized to run only when input values change. Impure pipes can have performance implications if not used carefully.  Custom pipe for changing the time. |
| 34. | What is an observable in Angular? |
| A. | In Angular, an Observable is a mechanism for handling asynchronous data streams. It represents a stream of data that can emit multiple values over time. Observables can be used for tasks like handling HTTP requests, events handling, and more. |
| 35. | What is RxJS in Angular? |
| A. | RxJS (Reactive extensions for JavaScript) is a library programming in Java script.It is widely used in Angular for handling asynchronous operations and managing data streams. RxJS provides a rich set of operators and functions that enable declarative and efficient handling of asynchronous events. |
| 36. | What is AOT in Angular? |
| A. | AOT stands for Ahead of time compilation in Angular. It is a compilation process that compiles Angular templates and components into highly efficient Java Script code during the build phase, before the application is deployed to the browser. AOT compilation improves Angular application performance by allowing the browser to directly download and execute optimized code, resulting in faster initial loading and improved overall performance. |
| 37. | What is JIT in Angular? |
| A. | JIT (Just-in-Time) compilation in Angular responsible for dynamically converts Angular templates into Java Script code as the app loads in the browser. It enables fast development with immediate changes and no separate build step. However, it leads to larger bundle sized and slightly slower initial load times. |
| 38. | What is subscription class in RxJS? |
| A. | The subscription class in RxJS Angular is used to manage the lifecycle of observables. It allows you to subscribe to an observable and unsubscribe from it when you no longer need to receive its emitted values. By unsubscribing, you prevent memory leaks and unnecessary computations. |
| 39. | What is BehaviourSubject class in RxJs? |
| A. | The BehaviourSubject class in RxJS Angular is a type of subject that represents a value that changes over time.  It keeps track of the current value and emits it to subscribers when they subscribe, as well as whenever the value changes. |
| 40. | What is Subject class in RxJS? |
| A. | The subject class in RxJS Angular is a type of observable that can be subscribed to and can also emit values. It acts as both an observable and an observer, allowing values to be multicast to multiple subscribers. |
| 41. | What is guard in Angular? |
| A. | In Angular, a guard is a feature that allows you to control access to certain routes of functionality within your application and preventing unauthorized access. |
| 42. | What is Resolve guard in Angular? |
| A. | In Angular, a Resolve guard is a feature that allows you to fetch data before activating a route. It helps in resolving data dependencies for a particular route and ensures that the route is activated only after the required data is loaded data. |
| 43. | What is ActivatedRoute service in Angular? |
| A. |  |
| 44. | What is canActivate in Angular? |
| A. |  |
| 45. | What is canDeactivate in Angular? |
| A. |  |
| 46. | What is @Input decorator in Angular? |
| A. |  |
| 47. | What is @Output decorator in Angular? |
| A. |  |
| 48. | What is EventEmitter class in Angular? |
| A. |  |
| 49. | What is @ViewChild decorator in Angular? |
| A. |  |
| 50. | What is @ContentChild decorator in Angular? |
| A. |  |
| 51. | What is @ContentChildren decorator in Angular? |
| A. |  |
| 52. | What is @ViewChilden decorator in Angular? |
| A. |  |
| 53. | What is @NgModule decorator in Angular? |
| A. |  |
| 54. | What is @HostBinding decorator in Angular? |
| A. |  |
| 55. | What is @HostListner decorator in Angular? |
| A. |  |
| 56. | What is common Module in Angular? |
| A. |  |
| 57. | What is Brower Module in Angular? |
| A. |  |
| 58. | What is HttpClient in Angular? |
| A. |  |
| 59. | What is HttpClientTesting Module in Angular? |
| A. |  |
| 60. | What is FormsModule in Angular? |
| A. |  |
| 61. | What is RouterModule in Angular? |
| A. |  |
| 62. | What is lazy-loading in Angular? |
| A. |  |
| 63. | What is changeDetectionStrategy in Angular? |
| A. |  |
| 64. | What is ngZone service in Angular? |
| A. |  |
| 65. | What is FormBuilder service in Angular? |
| A. |  |
| 66. | What is Rendere2 service in Angular? |
| A. |  |
| 67. | What is trackBy function in Angular? |
| A. |  |
| 68. | What is template reference variable in Angular? |
| A. |  |
| 69. | What is ElementRef class in Angular? |
| A. |  |
| 70. | What is viewEncapsulation in Angular? |
| A. |  |
| 71. | What is Angular Universal in Angular? |
| A. |  |
| 72. | What is Angular Ivy compiler in Angular? |
| 73. | What is view Engine in Angular? |
| A. |  |
| 74. | What is QueryList class in Angular? |
| A. |  |
| 75. | What is ActivateRouteSnapShot class in Angular? |
| A. |  |
| 76. | What is RouterStateSnapShot class in Angular? |
| A. |  |
| 77. | Learn Validators class in Angular? |
| A. |  |
| 78. | What is HttpHeaders class in Angular? |
| A. |  |
| 79. | What is HttpParams class in Angular? |
| A. |  |
| 80. | What is HttpInterceptor interface in Angular? |
| A. |  |
| 81. | What is component and Module archtechture in Angular? |
| A. |  |
| 82. | What is about Angular Architechture? |
| A. |  |
| 83. | What is FormArray class in Angular? |
| A. |  |
| 84. | What is NPM in Angular? |
| A. |  |
| 85. | What is Life cycle Hooks in Angular? |
| A. |  |
| 86. | What is Singleton Service in Angular? |
| A. |  |
| 87. | What is slice pipe in Angular? |
| A. |  |
| 88. | What are the similarities & differences between constructor & ngOnIt in Angular? |