**ANGULAR INTERVIEW QUESTIONS AND ANSWERS**

**1. What are the advantages of using Angular over other JavaScript frameworks?**

* Angular offers a complete solution for building web applications, including powerful features for building UIs, managing state, and handling data.
* Angular has a large and active community of developers, which means that there is a lot of documentation, resources, and support available.
* Angular has excellent performance and scalability, making it suitable for building complex applications that need to handle large amounts of data and user interactions.
* Angular has a modular architecture that allows developers to easily break their applications into smaller, more manageable parts.
* Angular uses TypeScript, which provides strong typing, better tooling, and improved error detection compared to vanilla JavaScript.

**2. What is AngularJS, and how does it differ from Angular?**

* AngularJS is an earlier version of Angular that was released in 2010. It is based on JavaScript and uses a different architecture and syntax compared to Angular.
* AngularJS is a monolithic framework, which means that it has a lot of built-in functionality that is tightly coupled together.
* AngularJS uses two-way data binding, which can make it difficult to manage state in larger applications.
* Angular, on the other hand, is a component-based framework that is built on top of TypeScript. It has a more modular architecture and uses a one-way data flow model that makes it easier to manage state and data.

**3. What are the differences between AngularJS and Angular?**

| **AngularJS** | **Angular** |
| --- | --- |
| Released in 2010 | Released in 2016 |
| Uses JavaScript for coding | Uses TypeScript for coding |
| Based on MVC (Model-View-Controller) architecture | Based on components architecture |
| Uses two-way data binding | Uses one-way data binding |
| Supports only desktop and laptop browsers | Supports mobile and desktop browsers |
| Does not support server-side rendering | Supports server-side rendering |
| AngularJS expressions are not safe to use | Angular expressions are safe to use |
| Limited in terms of performance and scalability | Offers better performance and scalability |
| Has a steep learning curve | Has a relatively flat learning curve |

**4. Explain the architecture of an Angular application.**

* An Angular application is made up of several key components, including modules, components, services, directives, and pipes.
* The application is organized into a tree structure, with a root module at the top and child modules, components, and services nested underneath it.
* Components are the building blocks of an Angular application and are responsible for managing the UI and user interactions.
* Services are used to provide data and functionality that can be shared across components.
* Directives and pipes are used to transform and manipulate data within the application.

**5. What are the main building blocks of an Angular application?**

* The main building blocks of an Angular application are modules, components, services, directives, and pipes.
* Modules provide a way to organize the application into separate functional areas and define dependencies.
* Components are the basic building blocks of the UI and are responsible for managing the presentation and user interactions.
* Services provide data and functionality that can be shared across components.
* Directives are used to manipulate the DOM and add behavior to elements.
* Pipes are used to transform and format data within the application.

**6. What is the difference between @ViewChild and @ContentChild in Angular?**

| **Aspect** | **@ViewChild** | **@ContentChild** |
| --- | --- | --- |
| Decorator | Used to get a reference to a child component or element within the current component’s view | Used to get a reference to a child component or element within the parent component’s content projection |
| Returns | The first matching child component or element | The first matching child component or element within a <ng-content> tag |
| Selector | Accepts a directive, component or element reference as a string or a type | Accepts a directive, component or element reference as a string or a type |
| Scope | Limited to the view of the component where it is defined | Applies to projected content of a component |
| Use cases | Accessing child components or elements to manipulate their behavior or properties | Accessing child components or elements that are projected into a parent component, to manipulate their behavior or properties |

**7. What is the difference between ng-content and ng-container in Angular?**

* ng-content is used to project the content of a component into the component’s template.
* ng-container is used as a placeholder for one or more structural directives, such as ngIf or ngFor.

**8. What is TypeScript, and why is it used in Angular?**

* TypeScript is a superset of JavaScript that adds static typing and other features to the language.
* TypeScript is used in Angular because it provides better tooling and improved error detection compared to vanilla JavaScript.
* TypeScript also helps to catch bugs early in the development process, which can save time and reduce the risk of errors in production.

**9. What are the key differences between Angular 2 and Angular 1?**

* Angular 2 is built on top of TypeScript, whereas Angular 1 is based on JavaScript.
* Angular 2 uses a component-based architecture, while Angular 1 uses a controller-based architecture.
* Angular 2 uses a one-way data flow model, while Angular 1 uses two-way data binding.
* Angular 2 has better performance and scalability than Angular 1.

**10. Explain the role of a component in an Angular application.**

* A component is the basic building block of an Angular application and is responsible for managing the presentation and user interactions.
* Components are made up of three main parts: the template, which defines the UI, the class, which provides the logic and data for the component, and the metadata, which provides additional information about the component.
* Components can communicate with other components using input aand output properties, and they can also emit events to communicate with the parent component or other components.

**11. What are the Components in Angular?**

In an Angular application, components serve as the fundamental elements of the user interface. Each component comes with a template and forms a subset of directives. The AppComponent, known as the root component, is typically present in every Angular application, and other components branch out from it, creating a hierarchy.

**12. What is the difference between a template reference variable and a component property in Angular?**

| **Aspect** | **Template reference variable** | **Component property** |
| --- | --- | --- |
| Usage | Used to access a DOM element or directive from a template | Used to hold and manipulate data within a component class |
| Declaration | Declared using the # symbol and a name in a template | Declared as a property within a component class |
| Accessibility | Accessible only within the template where it is declared | Accessible across the entire component class |
| Type | Can be of any type, depending on the element or directive it references | Can be of any type, depending on the data it holds |
| Value assignment | Assigned a value when the template is rendered | Assigned a value programmatically within the component class |
| Example | <input #myInput type="text"> where myInput is the reference variable | myProperty: string = "Hello World" where myProperty is the component property |

**13. What is data binding in Angular?**

* Data binding is a mechanism in Angular that allows you to synchronize the data between the UI and the component.
* There are three types of data binding in Angular: interpolation, property binding, and event binding.
* Interpolation allows you to insert values from the component into the UI.
* Property binding allows you to bind a property of an element to a property of the component.
* Event binding allows you to bind an event of an element to a method of the component.

**14. What is the difference between one-way and two-way data binding in Angular?**

* One-way data binding means that the data flows in one direction, from the component to the UI or vice versa. Changes made in the UI do not affect the component data.
* Two-way data binding means that the data flows in both directions, from the component to the UI and vice versa. Changes made in the UI are reflected in the component data and vice versa.

**15. What is an Angular module, and what is its purpose?**

* An Angular module is a container for a set of related components, services, directives, and pipes.
* The purpose of an Angular module is to organize the application into separate functional areas and define dependencies.
* Modules can be nested to create a hierarchy of functionality and can be lazy-loaded to improve application performance.

**16. What are Pipes in Angular?**

In Angular, pipes are functions that receive an input value and return a transformed output value. While Angular provides various built-in pipes, you can also create custom pipes that suit your requirements.

The following are some essential features of pipes:

* The pipe symbol “|” is used to define pipes.
* Pipes can be combined with other pipes to form a chain.
* Arguments can be passed to pipes by using the colon (:) symbol.

**17. Explain the Angular CLI and its features.**

* The Angular CLI is a command-line interface tool that helps developers create, manage, and deploy Angular applications.
* It provides a set of commands for creating new components, services, and modules, as well as generating code scaffolding, running tests, and building production-ready applications.
* Some of its features include:
  + Generating components, services, directives, and other Angular artifacts
  + Running a development server for testing and debugging
  + Building production-ready applications with optimized bundles
  + Running unit and end-to-end tests with the built-in testing framework
  + Integrating with popular build tools like Webpack and Bazel
  + Updating dependencies and keeping the project up-to-date with the latest Angular releases.

**18. What are the advantages of Angular over React?**

| **Angular** | **React** |
| --- | --- |
| Complete framework with a set of predefined tools and features | Lightweight library, requires additional libraries for certain functionalities |
| Supports two-way data binding | Supports only one-way data binding |
| Provides built-in dependency injection | Does not provide built-in dependency injection |
| Offers better performance for large-scale applications | Performance can degrade for large-scale applications |
| Provides better error handling and debugging tools | Error handling and debugging tools are not as extensive as Angular |
| Supports server-side rendering out-of-the-box | Requires additional setup for server-side rendering |
| Offers a more structured and opinionated approach to development | Offers a more flexible and unopinionated approach to development |
| Provides better support for typescript | Supports typescript but not as extensively as Angular |

**19. What is a directive in Angular?**

* A directive in Angular is a class that adds behavior to a DOM element, component, or attribute.
* Directives can be used to manipulate the DOM, add or remove elements, change styling, handle events, and communicate with other components and services.
* There are three types of directives in Angular:
  + Component directives: create reusable UI components that encapsulate logic and presentation.
  + Attribute directives: change the behavior or appearance of an existing element, such as adding a tooltip or highlighting text.
  + Structural directives: modify the layout or structure of the DOM, such as repeating a template or conditionally displaying elements.

**20. What is the difference between ngOnChanges and ngDoCheck in Angular?**

| **Aspect** | **ngOnChanges** | **ngDoCheck** |
| --- | --- | --- |
| Usage | Used to detect and respond to changes to input properties of a component | Used to detect and respond to changes to any component property or state that may not be detected by other lifecycle hooks |
| Triggering | Triggered every time an input property of the component changes | Triggered during every change detection cycle |
| Arguments | Accepts a SimpleChanges object that contains information about the changed input properties | Does not accept any arguments |
| Performance | Can impact performance if used with complex objects or arrays | Can impact performance if used incorrectly, as it is called frequently |
| Use cases | Used to perform specific actions in response to changes in input properties | Used to detect and respond to changes in component state or properties that are not detectable by other lifecycle hooks, such as changes in objects or arrays |
| Example | ngOnChanges(changes: SimpleChanges) { // do something } | ngDoCheck() { // do something } |

**21. Explain the difference between structural and attribute directives in Angular.**

* Attribute directives modify the behavior or appearance of an existing DOM element, using an HTML attribute.
* Structural directives modify the layout or structure of the DOM by adding or removing elements, using an HTML attribute prefixed with an asterisk (\*).
* Structural directives include ngIf, ngFor, and ngSwitch, which conditionally add or remove elements based on a condition or loop over a collection of data.
* Attribute directives include ngStyle, ngClass, and ngModel, which change the style, class, or value of an existing element.

**22. How would you use a directive to hide an element from the DOM in Angular?**

* To hide an element from the DOM using a directive in Angular, you can use the ngIf directive with a condition that evaluates to false.
* For example, to hide a div element based on a Boolean property in the component, you can use the following code:

<div \*ngIf=“hideElement”>This element is hidden</div>

* When the hideElement property is false, the element is not added to the DOM.

**23. What is an Angular service, and what is its purpose?**

* An Angular service is a class that provides functionality to other parts of the application, such as components, directives, and other services.
* Services can be used to share data between components, perform HTTP requests, manage state, handle authentication, and perform other tasks that are not related to the UI or presentation logic.
* Services are typically singleton instances, meaning that there is only one instance of the service per application, and they can be injected into other components or services using the Angular dependency injection system.

**24. What is the difference between ngFor and ngIf in Angular?**

| **Aspect** | **ngFor** | **ngIf** |
| --- | --- | --- |
| Usage | Used to render a template for each item in a collection | Used to conditionally render a template based on a boolean expression |
| Input | Accepts an iterable as input | Accepts a boolean expression as input |
| Output | Does not have any output | Emits a false value when the condition is false |
| Performance | Can impact performance when used with large collections | Can improve performance by not rendering unnecessary elements |
| DOM manipulation | Adds or removes elements from the DOM based on the size of the collection | Adds or removes elements from the DOM based on the boolean expression |
| Use cases | Used to iterate over a collection and render a template for each item | Used to conditionally render a template based on a boolean expression, such as hiding a section of a form or displaying an error message |
| Example | <div \*ngFor="let item of itemList"> {{ item }} </div> | <div \*ngIf="showContent"> This content will be displayed if showContent is true </div> |

**25. Explain the difference between a singleton and a non-singleton service in Angular.**

* A singleton service in Angular is a service that is instantiated only once per application and shared across all components and services that use it.
* A non-singleton service, on the other hand, is instantiated every time it is injected into a component or service that requires it.
* Singleton services are useful for sharing state between components and maintaining a single source of truth, while non-singleton services are useful for encapsulating behavior that is specific to a component or a set of components.

**26. What is an Angular pipe, and what is its purpose?**

* An Angular pipe is a feature that transforms the input data into a desired format or value that can be displayed in the UI.
* Pipes can be used to filter, sort, format, and transform data, and they can be chained together to perform complex operations.
* Pipes are used in the template syntax by appending the | character followed by the name of the pipe and any parameters.
* For example, the built-in uppercase pipe can be used to convert a string to all uppercase letters, like this: {{ myString | uppercase }}

**27. What are Single Page Applications (SPA)?**

Single page applications (SPAs) are web applications that load only once and require minimal changes to the user interface for new functionality. Unlike traditional web applications, they do not load new HTML pages to display new content, but instead dynamically generate it using JavaScript to manipulate the current page’s DOM components. The SPA approach provides faster response times and a more seamless user experience.

**28. What is the difference between ngOnInit and ngAfterViewInit in Angular?**

| **Aspect** | **ngOnInit** | **ngAfterViewInit** |
| --- | --- | --- |
| Timing | Lifecycle hook called after the constructor is executed and after the input properties have been set | Lifecycle hook called after the view and child views are initialized |
| Use case | Used to initialize properties or perform any setup logic that depends on input values | Used to access and manipulate the DOM after it has been initialized |
| Input/Output | Does not have any input or output properties | Does not have any input or output properties |
| Access | Can access input properties and instance variables | Can access the view and child views, as well as the DOM elements |
| Timing Example | Use ngOnInit to set up any default values for the component or to initialize any properties that rely on input values. | Use ngAfterViewInit to access and manipulate the DOM, such as setting focus on a form field or manipulating the styles of an element. |
| Calling Order | ngOnChanges -> ngOnInit -> ngDoCheck -> ngAfterContentInit -> ngAfterContentChecked -> ngAfterViewInit -> ngAfterViewChecked | N/A (Called only once after the view is initialized) |

**29. How would you create a custom pipe in Angular?**

* To create a custom pipe in Angular, you can use the @Pipe decorator to define a new class that implements the PipeTransform interface.
* The PipeTransform interface requires the implementation of a single method called transform, which takes an input value and an optional set of arguments and returns the transformed value.
* For example, to create a custom pipe that appends a given string to the input value, you can define a class like this:

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

@Pipe({ name: ‘append’ })  
export class AppendPipe implements PipeTransform {  
transform(value: string, suffix: string = ”): string {  
return value + suffix;  
}  
}

* This pipe can be used in the template syntax like this: {{ myString | append: ' - suffix' }}

**30. What are Angular guards, and how do they work?**

* Angular guards are a set of classes that control access to routes and prevent unauthorized access to protected pages.
* There are four types of guards in Angular:
  + CanActivate: controls whether a route can be activated and navigated to
  + CanActivateChild: controls whether child routes can be activated and navigated to
  + CanDeactivate: controls whether a user can leave a route and discard changes
  + CanLoad: controls whether a module can be loaded asynchronously
* Guards work by intercepting the route navigation process and checking whether the user is authorized to access the requested route.
* Guards can return a boolean value, a promise, or an observable to indicate whether the navigation should proceed or be blocked.

**31. What is the difference between a route parameter and a query parameter in Angular?**

| **Aspect** | **Route Parameter** | **Query Parameter** |
| --- | --- | --- |
| URL Syntax | Part of the URL path, denoted with a colon (:) followed by the parameter name | Part of the URL query string, denoted by a question mark (?) followed by the parameter name and value |
| Example URL | /users/:id | /users?id=123 |
| Usage | Used to pass information related to the requested resource | Used to pass additional information, such as filtering or sorting criteria |
| Access | Accessed using the ActivatedRoute service | Accessed using the ActivatedRoute service or the Router service |
| Observable | Accessed using the paramMap observable | Accessed using the queryParamMap observable |
| Multiple Values | Not supported. Only one value can be passed per parameter | Supported. Multiple parameters can be passed using the same parameter name |
| Encoding | URL encoded by default. Special characters are encoded using percent encoding | URL encoded by default. Special characters are encoded using percent encoding |
| Security | More secure than query parameters as the parameter values are not visible in the URL | Less secure than route parameters as the parameter values are visible in the URL |

**32. Explain the role of an Angular resolver.**

* An Angular resolver is a feature that preloads data for a route before the route is activated.
* Resolvers are used to prevent the UI from rendering until the necessary data is available, which can improve the user experience and reduce unnecessary HTTP requests.
* A resolver is a class that implements the Resolve interface and defines a resolve method that returns a value, a promise, or an observable that resolves to the required data.
* Resolvers are added to the route definition as a property called resolve, which maps the resolver to a named key that can be used in the component to retrieve the preloaded data.

**33. What is an Angular interceptor, and what is its purpose?**

* An Angular interceptor is a middleware function that can intercept HTTP requests and responses.
* Its purpose is to add or modify headers, transform data, handle errors, and perform other tasks related to HTTP communication.

**34. What is dependency injection, and how is it used in Angular?**

* Dependency injection is a design pattern in which an object’s dependencies are provided externally rather than being created inside the object.
* In Angular, it is used to provide services, which are injected into components and other services as needed.

**35. Explain the difference between providers and services in Angular.**

* A provider is a recipe for creating a service, while a service is an instance of a provider.
* Providers can be defined at the component or module level, while services are typically defined at the module level and shared across components.

**36. How would you create a custom provider in Angular?**

* To create a custom provider, you would define a class that implements the necessary methods for creating the service.
* You would then register the provider with the module or component by adding it to the providers array in the @NgModule or @Component decorator.

**37. What is a template in Angular, and how is it used?**

* A template is a piece of HTML code that defines the structure and content of a component.
* It is used to define how a component should look and behave based on its data and state.

**38. What is the difference between a module and a component in Angular?**

| **Aspect** | **Module** | **Component** |
| --- | --- | --- |
| Purpose | Acts as a container for related components, services, and other code | Represents a part of the user interface that controls the layout, styling, and behavior of a specific view |
| File Extension | .module.ts | .component.ts |
| Metadata Decorator | @NgModule | @Component |
| Selector | No selector | Unique selector used to identify the component in the HTML |
| Template | No template | Contains the HTML, CSS, and other view-related code |
| Styles | Can have global styles or scoped styles | Can have component-specific styles or no styles |
| Inputs | Can export components, directives, and pipes for use in other modules | Can have inputs to accept data from a parent component |
| Outputs | Can export services for use in other modules | Can have outputs to emit events to a parent component |
| Dependencies | Can import other modules, services, and dependencies | Can use services and dependencies provided by the module or parent component |
| Instantiation | Instantiated by the Angular compiler and runtime | Instantiated by the parent component or a routing configuration |
| Hierarchical Relationship | Can have a hierarchical relationship with other modules | Can have a hierarchical relationship with other components within the same view |
| Examples | FormsModule, HttpClientModule, BrowserModule | LoginComponent, HomeComponent, ProductListComponent |

**39. How would you use a template reference variable in Angular?**

* To use a template reference variable, you would define it using the # symbol and a name in the HTML template.
* You could then reference the variable in code using the @ViewChild decorator or by passing it as an argument to a method.

**40. Explain the difference between ngIf and ngSwitch in Angular.**

* ngIf is a structural directive that conditionally adds or removes elements from the DOM based on a boolean expression.
* ngSwitch is a structural directive that conditionally selects one of several elements to display based on a value.

**41. What is ngFor in Angular, and how is it used?**

* ngFor is a structural directive that repeats a block of HTML code for each item in an array.
* It is used to display a list of items or to generate multiple instances of a component.

**42. How would you use ngClass and ngStyle in Angular?**

* ngClass is a directive that allows you to dynamically add or remove CSS classes from an element based on an expression.
* ngStyle is a directive that allows you to dynamically set inline styles on an element based on an expression.

**43. What is the purpose of the async pipe in Angular?**

* The async pipe is a built-in Angular pipe that subscribes to an Observable or Promise and asynchronously retrieves and displays its value in the template.
* It simplifies the process of working with asynchronous data by automatically subscribing and unsubscribing to the data source.

**44. What is the Angular router, and how is it used?**

* The Angular router is a built-in library that provides navigation and routing functionality for Angular applications.
* It is used to define the routes for different pages and to navigate between them.

**45. What is the difference between a directive and a component in Angular?**

| **Directive** | **Component** |
| --- | --- |
| Used to add behavior or manipulate DOM elements | Used to define views or reusable pieces of UI |
| Can be used alone or with other directives | Typically used with other components and directives to create a larger application |
| Directives don’t have templates | Components have templates |
| Directives can be structural (changes the structure of the DOM) or attribute (changes the appearance of DOM elements) | Components are always structural and define their own custom elements |
| Examples: ngIf, ngFor, ngStyle | Examples: AppComponent, HeaderComponent, ProductListComponent |

**46. What is lazy loading, and how is it implemented in Angular?**

* Lazy loading is a technique for loading parts of an application only when they are needed, rather than loading the entire application upfront.
* In Angular, it is implemented using the loadChildren property in the routing configuration.

**47. How would you handle errors in an Angular application?**

* Error handling in Angular can be done using try-catch blocks, error handling services, or interceptors.
* The ErrorHandler class can also be used to handle uncaught exceptions and display an error message to the user.

**48. What is Angular’s ngTemplateOutlet and how does it work?**

* ngTemplateOutlet is used to dynamically render a template in the current component’s template.
* To use ngTemplateOutlet, add it to the parent component’s template and bind it to a template reference variable.
* The ngTemplateOutlet will render the template referenced by the variable in place of the `ngTemplate

**49. Explain the role of zone.js in an Angular application.**

* zone.js is a library that provides a way to track and manage asynchronous operations in an Angular application.
* It is used to detect changes in the application state and trigger change detection to update the UI.

**50. What is an Angular decorator, and how is it used?**

* An Angular decorator is a function that can modify the behavior of a class, method, or property.
* It is used to add metadata to a class or its members, such as the selector for a component or the input/output properties.

**51. How would you use the @ViewChild decorator in Angular?**

* The @ViewChild decorator is used to get a reference to a child component or element in the template.
* It can be used to access child components or elements from the parent component, and to interact with their properties or methods.

**52. What is the difference between a static and dynamic component in Angular?**

|  | **Static Component** | **Dynamic Component** |
| --- | --- | --- |
| Definition | A component with static data that does not change during runtime | A component with dynamic data that can change during runtime |
| Data Binding | One-way data binding | Two-way data binding |
| Rendering | Rendered once during the initial load | Rendered multiple times during the runtime |
| Input | Input data is passed via component properties during instantiation | Input data can be changed during runtime using property binding |
| Output | Output events cannot be emitted | Output events can be emitted to notify parent components or services of changes |
| Template | Template is static and does not change during runtime | Template can be dynamically modified using structural directives like ngIf and ngFor |
| Examples | Headers, footers, navigation menus | Search boxes, form inputs, real-time data displays |

**53. What is an Angular animation, and how is it implemented?**

* An Angular animation is a way to add visual effects to an application by animating the properties of HTML elements.
* It is implemented using the @angular/animations module, which provides a set of animation functions and timing functions.

**54. How would you create a custom Angular animation?**

* To create a custom Angular animation, you would define a trigger function that specifies the animation properties and timing.
* You would then use the trigger function in the template to apply the animation to an HTML element.

**55. What are Angular templates, and how are they used?**

* Angular templates are HTML files that define the structure and content of a component.
* They are used to display data and to define the behavior of the component based on its state.

**56. What is the difference between ng-template and ng-container in Angular?**

* ng-template is an Angular directive that defines a template that can be used to display data without rendering it.
* ng-container is an Angular directive that provides a way to group elements without creating an additional element in the DOM.
* ng-container can be used to wrap multiple elements inside an ngIf or ngFor directive, while ng-template is used to define a template for a directive or a component.

**57. Explain the role of NgModule in Angular.**

* NgModule is a core building block of Angular applications that defines a cohesive set of components, services, and directives that can be imported and used throughout the application.
* Each Angular application has at least one root module, which is responsible for bootstrapping the application and importing other modules as needed.
* NgModule provides a way to organize and modularize an application’s code, making it easier to maintain, test, and scale.

**58. What is the difference between a declarative and imperative approach in Angular?**

* In Angular, a declarative approach involves defining what you want to happen, and Angular takes care of the how.
* For example, when using templates to define a component’s view, you are declaring what the view should look like, and Angular takes care of rendering it.
* In contrast, an imperative approach involves manually writing the code to implement a desired behavior or feature.
* For example, when using the ViewChild decorator to get a reference to a child component in a parent component, you are using an imperative approach to access the child component.

**59. What are Angular forms, and how are they implemented?**

* Angular forms provide a way to collect and validate user input in an application.
* Forms can be implemented using either template-driven or reactive forms.
* Template-driven forms rely on directives in the HTML template to define form controls and validation rules, while reactive forms are built using TypeScript classes to define the form controls and validation logic.

**60. What is the difference between reactive and template-driven forms in Angular?**

* Reactive forms are built using TypeScript classes to define the form controls and validation logic, while template-driven forms rely on directives in the HTML template to define form controls and validation rules.
* Reactive forms are more flexible and offer better performance and scalability, while template-driven forms are simpler to implement and require less boilerplate code.

**61. How would you use form validation in an Angular application?**

* Form validation in Angular can be achieved by adding validation directives to the form controls, or by using custom validators defined in TypeScript classes.
* Validation directives can be used to validate input based on built-in or custom validation rules, such as required, minlength, and pattern.
* Custom validators can be created by defining a function that takes a form control as input and returns an object with validation errors if the input is invalid.

**62. Explain the difference between FormGroup and FormControl in Angular forms.**

* FormControl is a class that represents a single form control in Angular, such as an input field or checkbox.
* FormGroup is a class that represents a group of form controls, allowing you to group related form controls together and apply validation rules to the group as a whole.

**63. What is a dynamic component in Angular, and how is it used?**

* A dynamic component in Angular is a component that is created and added to the DOM at runtime, rather than being declared in the template.
* Dynamic components are useful for creating reusable and flexible components, such as modals or tooltips, that can be customized and added to different parts of the application.
* Dynamic components can be created using the ComponentFactoryResolver class, which dynamically creates a component factory based on a component type, and then uses that factory to create an instance of the component.

**64. How would you use lazy loading with dynamic components in Angular?**

* Lazy loading in Angular involves loading modules and components on demand, rather than upfront during application bootstrapping.
* To use lazy loading with dynamic components, you can create a module that exports the dynamic component, and then lazy load that module using the loadChildren property in the route definition.
* This allows the component to be loaded on demand only when the user navigates to the corresponding route.

**65. What is an Angular schematic, and how is it used?**

* An Angular schematic is a template for generating or modifying code in an Angular application.
* Schematics are used to automate common tasks and reduce boilerplate code, such as generating new components, services, or modules, or modifying existing code to conform to a certain coding style or architecture.
* Schematics can be created and shared by the community, allowing developers to easily extend and customize their Angular applications.

**66. What is a workspace in Angular, and what is its purpose?**

* An Angular workspace is a collection of projects and configuration files that define an Angular application.
* Workspaces provide a way to manage and organize large-scale applications, allowing developers to share code between projects and manage dependencies more efficiently.
* A workspace can contain one or more projects, each with its own configuration files and source code, and can be built and deployed as a single unit.

**67. How would you create an Angular library?**

* To create an Angular library, use the Angular CLI command ng generate library.
* This will generate a library project with a starter file and a configuration file.
* Add components, services, and modules to the library project.
* Build the library using the Angular CLI command ng build library-name.
* Publish the library to an npm registry or use it in another Angular project.

**68. Explain the role of TestBed in Angular testing.**

* TestBed is an Angular testing utility that provides a testing environment for Angular components, services, and directives.
* It creates a module that includes the component under test and its dependencies.
* TestBed can be used to create component instances, inject services, and access the component’s properties and methods.

**69. What is a Jasmine spy, and how is it used in Angular testing?**

* A Jasmine spy is a function that records calls made to it and allows you to test how it was called.
* In Angular testing, a spy can be used to replace a service or a component’s method with a fake one and verify that it was called with the correct arguments.
* Jasmine provides methods to create spies, such as spyOn and createSpy.

**70. What is Karma, and how is it used in Angular testing?**

* Karma is a test runner for JavaScript that is used to run tests in an actual browser or a headless browser.
* It is used in Angular testing to run unit tests and end-to-end tests.
* Karma can be configured to use various frameworks, such as Jasmine or Mocha, and can generate code coverage reports.

**71. How would you test an Angular component?**

* To test an Angular component, use the TestBed utility to create a testing environment and create an instance of the component.
* Use Jasmine to write test cases that verify the component’s behavior, such as testing the output of a method or verifying that a property was set correctly.
* Use the Angular testing API to interact with the component’s DOM, such as clicking a button or filling out a form.

**72. What is e2e testing, and how is it implemented in Angular?**

* End-to-end (e2e) testing is a type of testing that verifies the behavior of an application as a user would interact with it.
* In Angular, e2e testing is implemented using the Protractor framework, which uses Selenium WebDriver to automate browser interactions.
* E2e tests are written in TypeScript and use Jasmine syntax.

**73. What is the difference between beforeEach and beforeAll in Angular testing?**

* beforeEach is a Jasmine function that is called before each test case in a suite.
* beforeAll is a Jasmine function that is called once before all the test cases in a suite.
* beforeEach is used to set up the testing environment, such as creating a component instance or injecting a service.
* beforeAll is used to set up the testing environment once for all the test cases in a suite, such as connecting to a database or initializing a global variable.

**74. Explain the difference between a unit test and an integration test in Angular.**

* A unit test is a type of testing that verifies the behavior of a single unit of code, such as a method or a component.
* An integration test is a type of testing that verifies the behavior of multiple units of code working together, such as a component and its dependencies.
* In Angular, a unit test is typically written using TestBed to create a testing environment for the component or service under test.
* An integration test in Angular can be written using a combination of TestBed and Jasmine spy to verify the interaction between the components and their dependencies.

**75. How would you test an Angular service?**

* To test an Angular service, use the TestBed utility to create a testing environment and inject the service.
* Use Jasmine to write test cases that verify the service’s behavior, such as testing a method’s output or verifying that a property was set correctly.
* Use the Angular testing API to interact with the service’s dependencies, such as using a spy to verify that a method was called with the correct arguments.

**76. What is the purpose of a mock in Angular testing?**

* The purpose of a mock in Angular testing is to simulate the behavior of a dependency that a component or service depends on.
* It allows testing of the component or service in isolation, without needing to interact with the actual dependency.

**77. How would you mock an HTTP request in Angular testing?**

* In Angular testing, an HTTP request can be mocked using the HttpTestingController provided by the @angular/common/http/testing module.
* The HttpTestingController allows you to intercept and mock HTTP requests and responses in the test.

**78. Explain the role of a code coverage tool in Angular testing.**

* A code coverage tool in Angular testing is used to measure the percentage of code that is covered by the tests.
* It helps to identify areas of the code that are not covered by tests and to improve the overall quality of the code.

**79. What is the purpose of ngZone in Angular testing?**

* The purpose of ngZone in Angular testing is to track and manage asynchronous operations in the test.
* It allows the test to wait for asynchronous operations to complete before continuing with the next step.

**80. How would you use ngZone in an Angular test?**

* To use ngZone in an Angular test, you would inject the NgZone service into the test and use its run() method to execute code within the Angular zone.
* This allows the test to track and manage asynchronous operations, such as change detection and HTTP requests.

**81. What is Protractor, and how is it used for end-to-end testing in Angular?**

* Protractor is an end-to-end testing framework for Angular applications.
* It is used to simulate user interactions with the application, such as clicking buttons and entering text, and to verify that the application behaves as expected.

**82. Explain the difference between synchronous and asynchronous code in Angular.**

* Synchronous code in Angular is executed sequentially, one line at a time, and does not require waiting for any external operations to complete.
* Asynchronous code in Angular is executed concurrently and may require waiting for external operations to complete, such as HTTP requests or user input.

**83. How would you use Angular’s @HostBinding decorator?**

* @HostBinding is used to bind a property of a directive to a property of the host element.
* To use @HostBinding, import it from @angular/core and annotate the directive’s property with @HostBinding.
* Set the value of the directive’s property to the desired host element property.

**84. What is the purpose of the Promise object in Angular?**

* The purpose of the Promise object in Angular is to handle asynchronous operations and to return a value or an error once the operation is complete.
* It allows code to be written in a more concise and readable way than using callbacks.

**85. How would you use Observables in an Angular application?**

* Observables in Angular are used to handle asynchronous data streams, such as HTTP requests or user input.
* They can be used to handle events, to filter and transform data, and to manage the state of the application.

**86. What is Angular’s RouterOutlet and how does it work?**

* RouterOutlet is a directive that is used to render the component associated with the current route.
* To use RouterOutlet, add it to the parent component’s template and specify the location where the component should be rendered.
* When the route changes, RouterOutlet will dynamically create the component and insert it into the specified location.

**87. What is the difference between a hot and cold Observable in Angular?**

* A hot Observable in Angular is a stream that is shared among multiple subscribers and does not start over when a new subscriber is added.
* A cold Observable in Angular is a stream that is unique to each subscriber and starts over when a new subscriber is added.

**88. What is Angular’s HttpClientInterceptor and how does it work?**

* HttpClientInterceptor is used to intercept HTTP requests and responses and modify them before they are sent or received.
* To use HttpClientInterceptor, create a class that implements the HttpInterceptor interface and add it to the providers array in the app module.
* Implement the intercept() method to modify the request or response as needed.

**89. How would you use RxJS in an Angular application?**

* RxJS in Angular is used to handle asynchronous data streams using Observables and to provide a functional programming approach to handling data.
* It can be used to handle events, to filter and transform data, and to manage the state of the application.

**90. What is Angular’s FormBuilder and how does it work?**

* FormBuilder is a service that is used to create and manipulate Angular forms.
* To use FormBuilder, import it from @angular/forms and inject it into the component or service where the form will be used.
* Use the group() method to create a form group and the control() method to create form controls.

**91. Explain the difference between BehaviorSubject and ReplaySubject in RxJS.**

* BehaviorSubject and ReplaySubject are both types of RxJS Subjects.
* BehaviorSubject emits the most recent value to all subscribers, even if they subscribe after the value has been emitted.
* ReplaySubject records multiple values and emits them to subscribers based on a specified buffer size or time window.
* BehaviorSubject requires an initial value, whereas ReplaySubject does not.

**92. What is a Subject in RxJS, and how is it used?**

* A Subject is a type of RxJS Observable that allows for the multicast of values to multiple subscribers.
* It is used to share data between components or services in an Angular application.
* Subscribers to a Subject receive all emitted values, even if they subscribe after the value has been emitted.
* A Subject can emit values using its next() method.

**93. How would you create a custom operator in RxJS?**

* To create a custom operator in RxJS, you can use the pipe() method and return a function that takes an Observable as its input and returns a new Observable.
* The function should define the desired behavior of the operator, such as mapping or filtering the input Observable.
* The returned Observable can be subscribed to by any number of downstream Observers.

**94. What is NgRx, and how is it used in Angular?**

* NgRx is a state management library for Angular applications that uses the Redux pattern.
* It provides a centralized store for application state and facilitates communication between components and services.
* NgRx is used to manage complex application state, such as user authentication or data fetching.

**95. Explain the role of Redux in Angular.**

* Redux is a state management library that NgRx is based on.
* It provides a predictable, immutable state container for JavaScript applications.
* Redux helps manage application state in a consistent and reliable way, regardless of the size or complexity of the application.

**96. How would you implement state management in an Angular application?**

* State management in an Angular application can be implemented using a combination of services, RxJS Subjects, and the NgRx library.
* Services can be used to manage local state within a component, while RxJS Subjects can be used to share data between components or services.
* NgRx can be used for more complex state management, such as authentication or data fetching.

**97. What is the difference between NgRx and Redux in Angular?**

* NgRx is an implementation of the Redux pattern specifically for Angular applications.
* Redux is a more general state management library that can be used in a variety of JavaScript applications.

**98. What is the difference between a selector and a reducer in NgRx?**

* A selector is a function that retrieves a specific piece of state from the NgRx store.
* A reducer is a pure function that takes the current state and an action as input and returns a new state.

**99. How would you use NgRx to manage authentication in an Angular application?**

* NgRx can be used to manage authentication in an Angular application by storing the user’s authentication status and token in the store.
* Components can subscribe to the store to determine the user’s authentication status and take appropriate actions based on the user’s authorization.

**100. Explain the difference between a pure and impure pipe in Angular.**

* A pure pipe is a type of Angular pipe that does not change its output based on mutable inputs.
* An impure pipe is a type of Angular pipe that can change its output based on mutable inputs.
* Pure pipes are generally more performant and predictable, while impure pipes can cause unexpected behavior and performance issues.