Most Asked Angular Interview Questions

1. What is Angular? Why was it introduced?

Angular was introduced to create Single Page applications. This framework brings structure and consistency to web applications and provides excellent scalability and maintainability.

Angular is an open-source, JavaScript framework wholly written in TypeScript. It uses HTML's syntax to express your application's components clearly.

2. What is TypeScript?

TypeScript is a superset of JavaScript that offers excellent consistency. It is highly recommended, as it provides some syntactic sugar and makes the code base more comfortable to understand and maintain. Ultimately, TypeScript code compiles down to JavaScript that can run efficiently in any environment.

3. What is data binding? Which type of data binding does Angular deploy?

Data binding is a phenomenon that allows any internet user to manipulate Web page elements using a Web browser. It uses dynamic HTML and does not require complex scripting or programming. We use data binding in web pages that contain interactive components such as forms, calculators, tutorials, and games. Incremental display of a webpage makes data binding convenient when pages have an enormous amount of data.

Angular uses the two-way binding. Any changes made to the user interface are reflected in the corresponding model state. Conversely, any changes in the model state are reflected in the UI state. This allows the framework to connect the DOM to the Model data via the controller. However, this approach affects performance since every change in the DOM has to be tracked.

4. What are Single Page Applications (SPA)?

Single-page applications are web applications that load once with new features just being mere additions to the user interface. It does not load new HTML pages to display the new page's content, instead generated dynamically. This is made possible through JavaScript's ability to manipulate the DOM elements on the existing page itself. A SPA approach is faster, thus providing a seamless user experience.

5. Differentiate between Angular and AngularJS

The following table depicts the aspects of Angular vs AngularJS in detail:

Feature	AngularJS	Angular
Language	JavaScript	TypeScript
Architecture	Supports Model-View-Controller design	Uses components and directives
Mobile support	Not supported by mobile browsers	Supports all popular mobile browsers
Dependency Injection	Doesn't support	Supports
Routing	@routeProvider is used to provide routing information	@Route configuration is used to define routing information
Management	Difficult to manage with an increase in source code size	Better structured, easy to create and manage bigger applications

6. What are decorators in Angular?

Decorators are a design pattern or functions that define how Angular features work. They are used to make prior modifications to a class, service, or filter. Angular supports four types of decorators, they are:

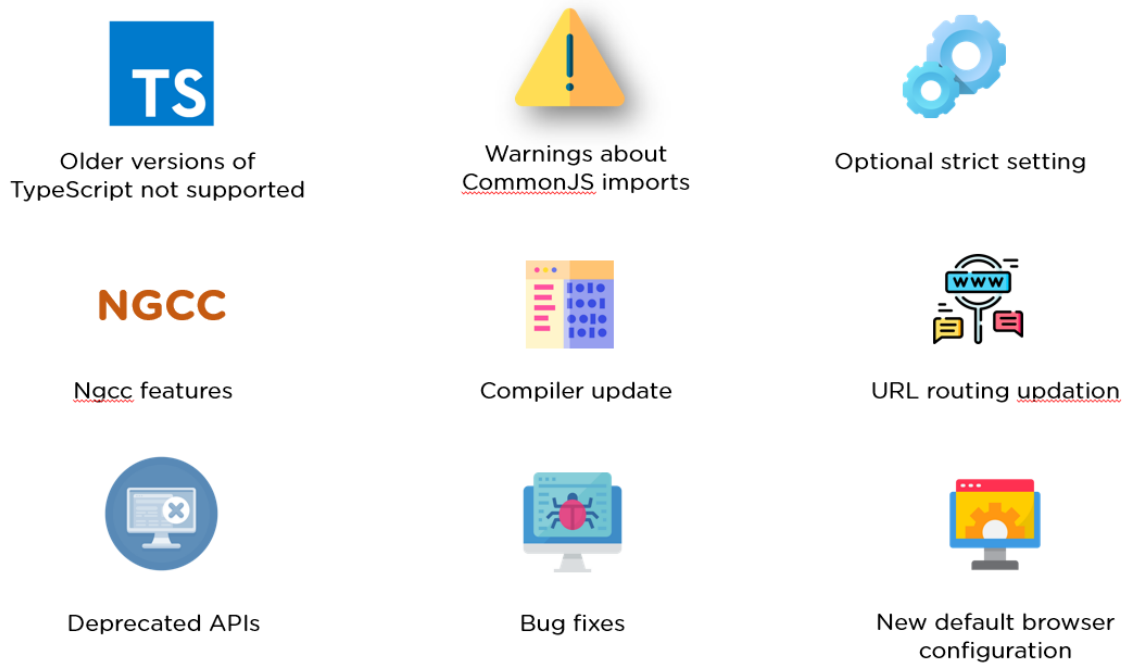
Class Decorators
Property Decorators
Method Decorators
Parameter Decorators

7. Mention some advantages of Angular.

Some of the common advantages of Angular are -

MVC architecture - Angular is a full-fledged MVC framework. It provides a firm opinion on how the application should be structured. It also offers bi-directional data flow and updates the real DOM.
Modules: Angular consists of different design patterns like components, directives, pipes, and services, which help in the smooth creation of applications.
Dependency injection: Components dependent on other components can be easily worked around using this feature.
Other generic advantages include clean and maintainable code, unit testing, reusable components, data binding, and excellent responsive experience.

8. What are the new updates with Angular10?



9. What are Templates in Angular?

Angular Templates are written with HTML that contains Angular-specific elements and attributes. In combination with the model and controller's information, these templates are further rendered to provide a dynamic view to the user.

10. What are Annotations in Angular?

Annotations in Angular are used for creating an annotation array. They are the metadata set on the class that is used to reflect the Metadata library.

Angular Interview Questions For Beginners

11. What are Directives in Angular?

Directives are attributes that allow the user to write new HTML syntax specific to their applications. They execute whenever the Angular compiler finds them in the DOM. Angular supports three types of directives.
Component Directives
Structural Directives
Attribute Directives

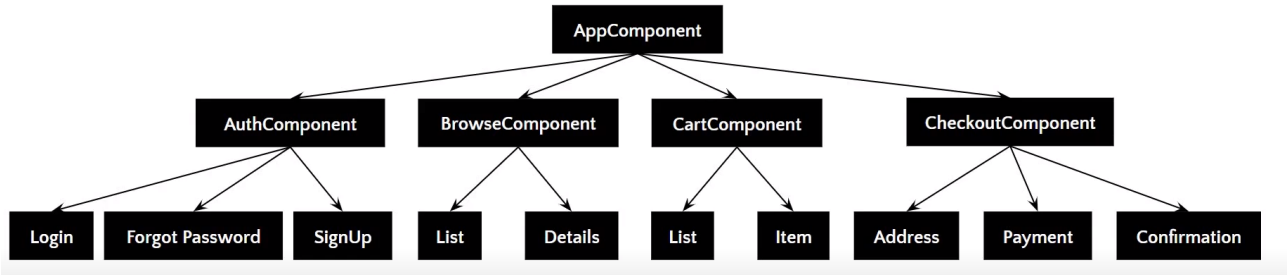
12. What is an AOT compilation? What are its advantages?

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

Some of its advantages are as follows.

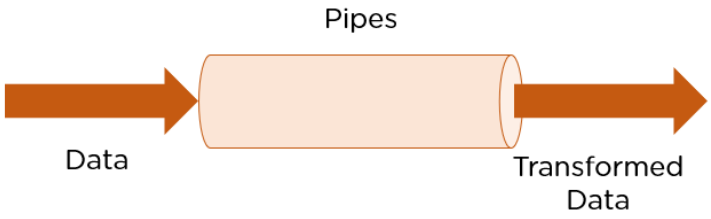
Faster rendering
Fewer asynchronous requests
Smaller Angular framework download size
Quick detection of template errors
Better security

13. What are Components in Angular?



Components are the basic building blocks of the user interface in an Angular application. Every component is associated with a template and is a subset of directives. An Angular application typically consists of a root component, which is the AppComponent, that then branches out into other components creating a hierarchy.

14. What are Pipes in Angular?



Pipes are simple functions designed to accept an input value, process, and return as an output, a transformed value in a more technical understanding. Angular supports several built-in pipes. However, you can also create custom pipes that cater to your needs.

Some key features include:

Pipes are defined using the pipe "|" symbol.
Pipes can be chained with other pipes.
Pipes can be provided with arguments by using the colon (:) sign.

15. What is the PipeTransform interface?

As the name suggests, the interface receives an input value and transforms it into the desired format with a transform() method. It is typically used to implement custom pipes.

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

@Pipe({

  name: 'demopipe'

})

export class DemopipePipe implements PipeTransform {

  transform(value: unknown, ...args: unknown[]): unknown {

    return null;

  }

}
```

16. What are Pure Pipes?

These pipes are pipes that use pure functions. As a result of this, a pure pipe doesn't use any internal state, and the output remains the same as long as the parameters passed stay the same. Angular calls the pipe only when it detects a change in the parameters being passed. A single instance of the pure pipe is used throughout all components.

17. What are Impure Pipes?

For every change detection cycle in Angular, an impure pipe is called regardless of the change in the input fields. Multiple pipe instances are created for these pipes. Inputs passed to these pipes can be mutable.

By default, all pipes are pure. However, you can specify impure pipes using the pure property, as shown below.

```
@Pipe({

  name: 'demopipe',

  pure : true/false

})

export class DemopipePipe implements PipeTransform {
```

18. What is an ngModule?

NgModules are containers that reserve a block of code to an application domain or a workflow. @NgModule takes a metadata object that generally describes the way to compile the template of a component and to generate an injector at runtime. In addition, it identifies the module's components, directives, and pipes, making some of them public, through the export property so that external components can use them.

19. What are filters in Angular? Name a few of them ?

Filters are used to format an expression and present it to the user. They can be used in view templates, controllers, or services. Some inbuilt filters are as follows.

date - Format a date to a specified format.
filter - Select a subset of items from an array.
Json - Format an object to a JSON string.
limitTo - Limits an array/string, into a specified number of elements/characters.
lowercase - Format a string to lowercase.

20. What is view encapsulation in Angular?

View encapsulation defines whether the template and styles defined within the component can affect the whole application or vice versa. Angular provides three encapsulation strategies:

Emulated - styles from the main HTML propagate to the component.
Native - styles from the main HTML do not propagate to the component.
None - styles from the component propagate back to the main HTML and therefore are visible to all components on the page.

21. What are controllers?

AngularJS controllers control the data of AngularJS applications. They are regular JavaScript Objects. The ng-controller directive defines the application controller.

22. What do you understand by scope in Angular?

The scope in Angular binds the HTML, i.e., the view, and the JavaScript, i.e., the controller. It as expected is an object with the available methods and properties. The scope is available for both the view and the controller. When you make a controller in Angular, you pass the $scope object as an argument.

23. Explain the lifecycle hooks in Angular

In Angular, every component has a lifecycle. Angular creates and renders these components and also destroys them before removing them from the DOM. This is achieved with the help of lifecycle hooks. Here's the list of them -

ngOnChanges() - Responds when Angular sets/resets data-bound input properties.
ngOnInit() - Initialize the directive/component after Angular first displays the data-bound properties and sets the directive/component's input properties/
ngDoCheck() - Detect and act upon changes that Angular can't or won't detect on its own.
ngAfterContentInit() - Responds after Angular projects external content into the component's view.
ngAfterContentChecked() - Respond after Angular checks the content projected into the component.
ngAfterViewInit() - Respond after Angular initializes the component's views and child views.
ngAfterViewChecked() - Respond after Angular checks the component's views and child views.
ngOnDestroy - Cleanup just before Angular destroys the directive/component.

24. What is String Interpolation in Angular?

String Interpolation is a one-way data-binding technique that outputs the data from TypeScript code to HTML view. It is denoted using double curly braces. This template expression helps display the data from the component to the view.

{{ data }}

25. What are Template statements?

Template statements are properties or methods used in HTML for responding to user events. With these template statements, the application that you create or are working on, can have the capability to engage users through actions such as submitting forms and displaying dynamic content.

For example,

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

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

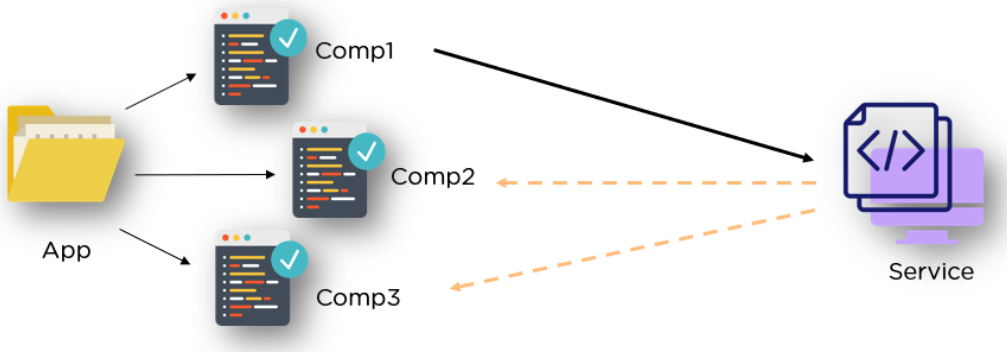
26. What is the difference between AOT and JIT?

Ahead of Time (AOT) compilation converts your code during the build time before the browser downloads and runs that code. This ensures faster rendering to the browser. To specify AOT compilation, include the --aot option with the ng build or ng serve command.

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.

27. What are Services in Angular?

Angular Services perform tasks that are used by multiple components. These tasks could be data and image fetching, network connections, and database management among others. They perform all the operational tasks for the components and avoid rewriting of code. A service can be written once and injected into all the components that use that service.



28. What is Eager and Lazy loading?

Eager loading is the default module-loading strategy. Feature modules under Eager loading are loaded before the application starts. This is typically used for small size applications.
Lazy loading dynamically loads the feature modules when there's a demand. This makes the application faster. It is used for bigger applications where all the modules are not required at the start of the application.

29. How does an Angular application work?

An Angular application is a Single Page Application, or SPA. This means that the entire application lives within a single page, and all of the resources (HTML, CSS, JavaScript, etc.) are loaded when the page is first loaded. Angular uses the Model-View-Controller, or MVC, architecture pattern to manage its data and views. The Model is the data that the application uses, the View is what the user sees, and the Controller is responsible for managing communication between the Model and the View.
When a user interacts with an Angular application, the Angular framework will automatically update the View to reflect any changes in the data. This means that Angular applications are very responsive and fast, as the user does not need to wait for the page to reload in order to see updated data.
Angular applications are also very scalable, as they can be divided into small modules that can be loaded independently of each other. This means that an Angular application can be easily extended with new functionality without having to rewrite the entire application.
Overall, Angular applications are very fast, responsive, and scalable. They are easy to develop and extend, and provide a great user experience.
The following is is an example of coding from an angular.json file:

```
 "build": {
       "builder": "@angular-devkit/build-angular:browser",
       "options": {
         "outputPath": "dist/angular-starter",
         "index": "src/index.html",
         "main": "src/main.ts",
         "polyfills": "src/polyfills.ts",
         "tsConfig": "tsconfig.app.json",
         "aot": false,
         "assets": [
           "src/favicon.ico",
           "src/assets"
         ],
         "styles": [
           "./node_modules/@angular/material/prebuilt-themes/deeppurple-amber.css",
           "src/style.css"
         ]
       }
     }
```

30. Explain components, modules and services in Angular.

***Component*:**

Components, modules and services are the three fundamental building blocks in Angular. Components are the smallest, self-contained units in an Angular application. They are typically used to represent a view or UI element, such as a button or a form input field.

Code example:

```
import { Component, OnInit } from '@angular/core';
    @Component({
      selector: 'app-test',
      templateUrl: './test.component.html',
      styleUrls: ['./test.component.css']
    })
    export lass TestComponent implements OnInit {
      constructor() {}
      ngOnInit() {
      }
    }
```

Modules:

Modules are larger units that group together one or more related components. Services are singleton objects that provide specific functionality throughout an Angular application, such as data access or logging.

Code example:

```
import { BrowserModule } from '@angular/platform-browser';
     import { NgModule } from '@angular/core';
     import { AppComponent } from './app.component';
     import { TestComponent } from './test/text.component';
     @NgModule({
       declarations: [
         AppComponent,
         TestComponent
       ],
       imports: [
         BrowserModule
       ],
       providers: [],
       bootstrap: [AppComponent]
     })
     export class AppModule { }
```

Each component in Angular has its own isolated scope. This means that a component's dependencies (services, other components, etc.) are not accessible to any other component outside of its own scope. This isolation is important for ensuring modularity and flexibility in an Angular application.

Services:
Services, on the other hand, are not isolated and can be injected into any other unit in an Angular application (component, module, service, etc.). This makes them ideal for sharing data or functionality across the entire app.
Code example:

```
import { Injectable } from '@angular/core';
    @Injectable({
      providedIn: 'root'
    })
    export class TestServiceService {
      constructor() { }
    }
```

When designing an Angular application, it is important to keep these three building blocks in mind. Components should be small and self-contained, modules should group together related components, and services should provide shared functionality across the entire app. By following this design principle, you can create an Angular application that is modular, flexible, and easy to maintain.

31. How do you share data between components in Angular?

Sharing data between components in Angular is simple and easy. To share data, all you need to do is use the Angular CLI to generate a new service. This service can be injected into any component and will allow the components to share data.
To generate a new service, use the following Angular CLI command:

- ng generate service my-data-service
- This will create a new service file called my-data-service.ts in the src/app folder.
- Inject the service into any component that needs to share data:
- import { MyDataService } from ‘./my-data.service';
- constructor(private myDataService: MyDataService) { }
- Once injected, the service will be available in the component as this.myDataService.
- To share data between components, simply use the setData() and getData() methods:
- this.myDataService.setData('some data’);
- const data = this.myDataService.getData();

1. What is Angular?
A TypeScript-based open-source framework for building web applications.

2. What are components in Angular?
Building blocks of Angular applications, consisting of HTML templates, CSS styles, and TypeScript logic.

3. What is a module in Angular?
A container for a cohesive block of code dedicated to an application domain, workflow, or set of capabilities.

4. Explain Angular CLI.
Command Line Interface tool to create, develop, and maintain Angular applications.

5. What are services in Angular?
Singleton objects to share data or logic across components.

6. What is TypeScript?
A typed superset of JavaScript that compiles to plain JavaScript.

7. Explain TypeScript interfaces.
Used to define the shape of an object, ensuring it adheres to a specific structure.

8. What are decorators in TypeScript?
Special kind of declaration used to attach metadata to class declarations and members.

9. What is data binding in Angular?
Mechanism to synchronize data between the component and the view.

10. Explain two-way data binding.
Combines property binding and event binding to sync data between the model and the view.

11. What is dependency injection?
A design pattern used to implement IoC, allowing a class to receive its dependencies from external sources.

12. Explain lifecycle hooks.
Methods that Angular calls at specific stages of a component's lifecycle (e.g., ngOnInit, ngOnDestroy).

13. How do you perform CRUD operations in Angular?
Using HTTP methods (GET, POST, PUT, DELETE) with Angular's HttpClient service.

14. Explain HttpClient in Angular.
A service for making HTTP requests in an Angular application.

15. How to handle errors in HttpClient?
Using the `***catchError***` operator in RxJS.

16. What is RxJS?
A library for reactive programming using Observables, used extensively in Angular.

17. What are the types of forms in Angular?
Template-driven and Reactive forms.

18. Explain template-driven forms.
Forms created using Angular directives in the template.

19. Explain reactive forms.
Forms created programmatically in the component using FormGroup and FormControl.

20. What is Angular Router?
A service to enable navigation between different views or components.

21. How do you define routes in Angular?
Using the RouterModule's `forRoot` method with an array of route configurations.

22. What is lazy loading in Angular?
Loading modules only when they are needed, reducing the initial load time.

23. What are route guards?
Interfaces that allow you to control navigation to and from a route.

24. What is state management?
Managing the state of an application consistently, especially in complex applications.

25. Explain NgRx.
A reactive state management library for Angular using RxJS.

26. What is unit testing?
Testing individual components or services in isolation.

27. What is end-to-end testing?
Testing the entire application flow from start to finish.

28. What tools are used for testing in Angular?
Jasmine for unit testing, Protractor for end-to-end testing.

29. How do you optimize Angular applications?

Lazy loading, AOT compilation, optimizing change detection, and using trackBy in ngFor.

30. What is AOT compilation?

Ahead-of-Time compilation that converts Angular HTML and TypeScript code into efficient JavaScript code during the build phase.

31. Explain the use of pipes in Angular.

Transforming data in templates.

32. What is Angular Universal?

A project that allows server-side rendering of Angular applications.

33. What is the difference between an Observable and a Promise?

Observables are lazy and can handle multiple values over time, while Promises are eager and handle a single value.

34. How do you handle asynchronous operations in Angular?

Using Observables with RxJS or Promises.

35. What is a directive in Angular?

A class that adds behavior to elements in the DOM.

36. Explain Angular Material.

A UI component library for Angular applications that follows Material Design guidelines.

37. What is the purpose of the Angular.json file?

Configuration file for Angular CLI projects, including build and serve configurations.

38. How do you implement authentication in Angular?

Using services to manage authentication logic, and route guards to protect routes.

39. What is the difference between ViewChild and ContentChild?

ViewChild is used to access a child component or directive from the component's view, while ContentChild is used to access projected content within the component.

40. How do you handle forms validation in Angular?

Using built-in validators and custom validators in template-driven or reactive forms.

1. What is CRUD?

CRUD stands for Create, Read, Update, and Delete, which are the basic operations for persistent storage.

2. How does Angular handle HTTP requests for CRUD operations?

Angular uses HttpClient module to make HTTP requests to a server for CRUD operations.

3. Explain Angular's HttpClientModule and its usage.

HttpClientModule is used to send HTTP requests and handle responses from a server in Angular applications.

4. What is a service in Angular?

A service in Angular is a class that encapsulates reusable logic and data, often used for handling business logic and data operations like CRUD.

5. How do you perform data binding in Angular for CRUD operations?

Angular uses [(ngModel)] for two-way data binding in forms, enabling easy binding of data to and from form fields.

6. Describe the steps to create a new component in Angular.

Use `ng generate component componentName` or its shorthand `ng g c componentName` to generate a new component.

7. What are Angular modules, and why are they used?

Angular modules (NgModule) are used to organize an application into cohesive blocks of functionality, including components, services, and other modules.

8. How do you handle routing in Angular for CRUD operations?

Angular Router module is used for navigation and routing between different components and views based on URLs.

9. Explain Angular forms and their types.

Angular forms can be Template-driven or Reactive forms. Template-driven forms use directives like ngModel for two-way data binding, while Reactive forms use FormBuilder for a more explicit approach.

10. What is Dependency Injection (DI) in Angular?

DI is a design pattern in Angular where components and services request dependencies from Angular's DI system instead of creating them directly.

11. How do you handle errors in Angular HTTP requests?

Use catchError operator in RxJS to handle errors in HTTP requests and provide fallback behavior or error messages.

12. What is Angular CLI, and why is it useful?

Angular CLI (Command Line Interface) is a powerful tool for initializing, developing, and maintaining Angular applications, providing commands for generating components, services, modules, etc.

13. Explain Angular decorators and provide examples.

Decorators in Angular are functions that modify the behavior of classes or properties. Examples include @Component, @Injectable, @Input, @Output, etc.

14. How do you pass data between components in Angular?

Use @Input() and @Output() decorators for parent-child component communication, and services or state management libraries (like NgRx or Angular services) for cross-component communication.

15. What is Angular's NgRx library, and why would you use it?

NgRx is a state management library for Angular applications based on Redux principles, useful for managing complex application state and data flow.

16. Explain Angular lifecycle hooks and provide examples.

Angular lifecycle hooks are methods that Angular calls at specific points in the life cycle of a component or directive. Examples include ngOnInit, ngOnDestroy, ngAfterViewInit, etc.

17. How do you handle authentication and authorization in Angular applications?

Use Angular's HttpClient to send authentication requests, store tokens in local storage or cookies, and implement guards (CanActivate, CanLoad) for authorization checks.

18. What are Angular directives, and how are they used?

Directives in Angular are markers on a DOM element that tell Angular to attach a specific behavior to that element or modify its appearance or structure.

19. Explain Angular's change detection strategy.

Angular uses change detection to automatically update the DOM when the application state changes. Strategies like OnPush can optimize performance by reducing unnecessary checks.

20. What are pipes in Angular, and why are they useful?

Pipes in Angular transform data for display in a template. They are useful for formatting dates, numbers, currency, and other data types.

21. How do you optimize Angular application performance?

Use lazy loading for modules, optimize change detection strategy, minimize HTTP requests, use production builds, and leverage server-side rendering for initial page load.

22. Describe Angular's testing support and tools.

Angular provides tools like TestBed for unit testing components, services, and modules. Jasmine and Karma are commonly used for testing Angular applications.

23. What is Angular Universal, and when would you use it?

Angular Universal is a technology for server-side rendering (SSR) of Angular applications, useful for improving performance, SEO, and initial page load times.

24. How do you handle form validation in Angular?

Angular provides built-in validators like required, minLength, maxLength, pattern, etc. Use FormGroup and FormControl to implement validation in forms.

25. Explain the role of TypeScript in Angular development.

TypeScript is a superset of JavaScript that adds static types, classes, and interfaces. It enhances code quality, provides better tooling, and improves maintainability in Angular projects.

26. What is lazy loading in Angular, and why is it beneficial?

Lazy loading is a technique where modules are loaded on demand rather than upfront. It reduces initial bundle size and improves application load time.

27. How do you deploy an Angular application to production?

Build the application using `ng build --prod`, then deploy the output files (usually found in the `dist` folder) to a web server or hosting service.

28. Explain Angular's AOT (Ahead-of-Time) compilation.

AOT compilation converts Angular HTML and TypeScript code into efficient JavaScript code during the build phase, improving application startup performance.

29. What are Angular animations, and how are they implemented?

Angular animations are used to create visually appealing transitions and effects in Angular applications. Implement them using Angular's BrowserAnimationsModule and @angular/animations module.

30. How do you handle internationalization (i18n) in Angular?

Use Angular's built-in i18n support to translate text and localize applications for different languages and cultures.

31. Describe the differences between Angular and AngularJS.

Angular is a complete rewrite of AngularJS, focusing on component-based architecture, improved performance, TypeScript support, and enhanced tooling.

32. What are Angular interceptors, and why are they useful?

Angular interceptors are used to intercept HTTP requests and responses. They are useful for adding headers, handling errors, or modifying requests before they are sent.

33. Explain Angular's component-based architecture.

Angular applications are built as a hierarchy of components, where each component encapsulates its own logic, template, and styling, promoting reusability and maintainability.

34. How do you handle dependency injection across different Angular modules?

Use Angular's providers array in NgModule metadata to provide services across modules, ensuring dependencies are injected where needed.

35. What are Angular templates and template expressions?

Angular templates are HTML views enhanced with Angular directives and binding markup. Template expressions are JavaScript-like expressions used within templates to display data dynamically.

36. How do you handle cross-site scripting (XSS) in Angular applications?

Use Angular's built-in sanitization mechanisms and safe HTML bindings like [innerHTML] or DomSanitizer to prevent XSS attacks.

37. Explain Angular's reactive programming approach with RxJS.

Angular leverages RxJS (Reactive Extensions for JavaScript) for reactive programming, enabling asynchronous data streams and event handling in applications.

38. What are Angular guards, and when are they used?

Angular guards (CanActivate, CanLoad, CanDeactivate, etc.) are used to control access to routes or prevent navigation based on certain conditions like authentication status or user permissions.

39. How do you handle asynchronous operations in Angular services?

Use Observables or Promises in Angular services to handle asynchronous operations such as HTTP requests or data fetching.

40. Describe the role of Angular decorators like @ViewChild and @ContentChild.

@ViewChild and @ContentChild decorators in Angular are used to access child components or elements within a parent component's template for interaction or manipulation.

JavaScript interview questions

1. What is JavaScript?
- JavaScript is a high-level, interpreted programming language used to create dynamic and interactive websites.

2. What are the data types in JavaScript?
- JavaScript has six primitive data types: string, number, boolean, null, undefined, and symbol (added in ES6), along with object which includes arrays and functions.

3. What is the difference between `==` and `===` in JavaScript?
- `==` checks for equality of value, whereas `===` checks for equality of value and type.

4. How do you declare variables in JavaScript?
- Variables in JavaScript are declared using `var`, `let`, or `const`.

5. What is the difference between `let`, `var`, and `const`?
- `var` has function-level scope, `let` has block-level scope (introduced in ES6), and `const` is used for constants whose value cannot be changed once defined.

6. How do you create functions in JavaScript?
- Functions in JavaScript can be created using function declarations, function expressions, arrow functions (introduced in ES6), and methods (functions defined on objects).

7. What are JavaScript callbacks?
- Callbacks are functions passed as arguments to another function to be executed later asynchronously.

8. What is an arrow function in JavaScript?
- Arrow functions are a concise way to write anonymous functions in JavaScript, introduced in ES6.

9. How does prototypal inheritance work in JavaScript?
- JavaScript uses prototypal inheritance where objects can inherit properties and methods from other objects.

10. What are closures in JavaScript?
- Closures are functions that retain access to variables from the scope in which they were defined, even after the parent function has finished executing.

11. Explain asynchronous programming in JavaScript.
- Asynchronous programming in JavaScript involves executing code asynchronously using callbacks, Promises (introduced in ES6), and async/await (introduced in ES8).

12. How do you handle errors in JavaScript?

- Errors in JavaScript can be handled using try...catch blocks to catch exceptions and handle them gracefully.

13. What is the DOM (Document Object Model) in JavaScript?

- The DOM is a programming interface for web documents that allows JavaScript to manipulate HTML and CSS dynamically.

14. How do you manipulate the DOM using JavaScript?

- DOM manipulation in JavaScript involves methods like getElementById, querySelector, innerHTML, appendChild, etc., to modify HTML elements.

15. What are events in JavaScript?

- Events in JavaScript are actions or occurrences (like clicks, mouse movements, key presses) that can be detected and used to trigger JavaScript code.

16. What are JavaScript promises?

- Promises are objects representing the eventual completion or failure of an asynchronous operation, allowing more readable and manageable async code.

17. How do you iterate over objects and arrays in JavaScript?

- Arrays can be iterated using loops like for and forEach, while objects can be iterated using for...in loop or Object.keys().

18. What are template literals in JavaScript?

- Template literals (introduced in ES6) are enclosed by backticks (` `) and allow embedded expressions and multiline strings.

19. What is strict mode in JavaScript?

- Strict mode is a way to opt-in to a restricted version of JavaScript that disallows certain features and provides more secure coding practices.

20. What is JSON and how is it related to JavaScript?

- JSON (JavaScript Object Notation) is a lightweight data-interchange format that is easy for humans to read and write, and easy for machines to parse and generate. It is based on JavaScript object syntax.

21. How do you check the type of a variable in JavaScript?

- The typeof operator is used to check the type of a variable or expression.

22. What are modules in JavaScript?

- Modules in JavaScript are reusable pieces of code that can be exported from one program and imported into another, helping to organize and structure code.