Multiple-Choice Questions

1. What type of framework is Angular?

a) Backend
b) Frontend
c) Full-stack
d) Middleware

2. Which programming language is Angular primarily built with?

a) JavaScript
b) TypeScript
c) Python
d) PHP

3. Angular was developed by which company?

a) Facebook
b) Microsoft
c) Google
d) Amazon

4. What does SPA stand for in the context of Angular?

a) Single Programming Application
b) Single Page Application
c) Simplified Programming Application
d) Secure Page Application

5. Which of the following is NOT a feature of Angular?

a) Component-based architecture
b) Two-way data binding
c) Dependency injection
d) Procedural programming

d) Procedural programming because it is not a feature of Angular's design, which focuses on a more modern, component-based, and declarative approach.

6. What is the primary architectural pattern used in Angular?

a) MVC
b) Component-based
c) MVVM
d) Singleton

7. What is the purpose of Angular CLI?

a) To compile JavaScript code
b) To provide command-line tools for Angular development
c) To debug Angular applications
d) To connect Angular to databases

8. What is the command to create a new Angular project using CLI?

a) ng new project-name
b) ng create project-name
c) ng build project-name
d) ng serve project-name

9. Which Angular feature handles navigation between components?

a) Directives
b) Services
c) Routing
d) Pipes

Angular Router is designed specifically to map URLs to components, allowing users to navigate through different views of an application. It provides features like lazy loading, route guards, and dynamic routing, enabling a seamless experience in SPAs. Therefore, routing is the Angular feature responsible for handling navigation between components.

10. What is the default port for Angular's development server?

a) 8080
b) 3000
c) 4200
d) 8000

11. Which directive is used for structural changes in the DOM?

a) ngClass
b) ngStyle
c) *ngIf
d) ngBind

12. Angular is a rewrite of which framework?

a) React
b) Vue.js
c) AngularJS
d) Backbone.js

13. Which of the following is a structural directive in Angular?

a) ngFor
b) ngModel
c) ngClass
d) ngBind

---

14. What is the purpose of RxJS in Angular?

a) For routing
b) For animations
c) For handling asynchronous data streams
d) For unit testing

15. Which lifecycle hook is triggered after the component is initialized?

a) ngOnInit
b) ngOnChanges
c) ngDoCheck
d) ngAfterViewInit

16. What command compiles Angular code for production?

a) ng build
b) ng serve
c) ng deploy
d) ng compile

17. What type of binding is used for event handling in Angular?

a) Property binding
b) Event binding
c) Two-way binding
d) Directive binding

18. Which of the following is a core module in Angular?

a) RouterModule
b) MaterialModule

c) BootstrapModule
d) HttpModule

19. Which file acts as the entry point for an Angular application?

a) main.ts
b) app.module.ts
c) index.html
d) style.css

20. What does AOT stand for in Angular?

a) Angular Optimization Tool
b) Ahead of Time
c) Asynchronous Object Transfer
d) Angular Object Tracking

Multiple-Choice Questions on Angular CLI

---

1. What is the primary purpose of Angular CLI?
a) To manage database connections
b) To provide a command-line interface for Angular development
c) To handle backend server operations
d) To compile JavaScript into TypeScript

---

2. Which command is used to install Angular CLI globally?
a) npm install angular-cli
b) npm install @angular/cli
c) npm install -g @angular/cli
d) install-angular-cli

---

3. How can you verify the installed version of Angular CLI?
a) ng verify
b) ng check
c) ng version
d) angular --version

---

4. What command is used to create a new Angular project?
a) ng init project-name
b) ng build project-name
c) ng new project-name
d) ng create project-name

5. By default, on which port does the Angular development server run?
a) 8080
b) 3000
c) 4200
d) 8000
6. Which command generates a new component in an Angular project?
a) ng create component component-name
b) ng generate component component-name
c) ng add component component-name
d) ng build component component-name
7. What does the ng build --prod command do?
a) Builds the app for local development
b) Compiles the app in debug mode
c) Creates an optimized production build
d) Builds the app and deploys it to the server
8. Which command is used to start the Angular development server?
a) ng build
b) ng serve
c) ng start
d) ng run

9. What command can you use to add a library or feature to your Angular project?
a) ng add
b) ng include
c) ng install
d) ng library

---

10. What command is used to run unit tests in an Angular application?
a) ng test
b) ng lint
c) ng serve --test
d) ng run-tests

20 Multiple-Choice Questions on Angular Components

1. What is a component in Angular?

a) A way to manage routes
b) A building block of the application's user interface
c) A method to handle HTTP requests
d) A part of the module system

2. Which decorator is used to define a component in Angular?

a) @Component
b) @Directive
c) @Service
d) @NgModule

3. What does the selector property in the @Component decorator define?

a) The HTML template of the component
b) The CSS styles of the component
c) The tag name used to include the component in templates
d) The data-binding syntax for the component

---

4. Which Angular CLI command is used to generate a new component?

a) ng new component-name
b) ng create component-name
c) ng generate component component-name
d) ng build component-name

5. What file contains the logic and metadata for a component?

a) .component.html
b) .component.css
c) .component.ts
d) .component.spec.ts

6. What is the default scope of component styles in Angular?

a) Global
b) Local to the component
c) Shared across the application
d) Overridden by parent components

7. How do you include a component in another component's template?

a) Use the component's class name
b) Use the component's selector as an HTML tag
c) Import the component's TypeScript file
d) Declare the component in angular.json

8. What is the purpose of the ngOnInit lifecycle hook?

a) Initialize the component’s metadata
b) Respond to input property changes
c) Perform initialization logic after the component is created
d) Clean up resources before the component is destroyed

9. Where do you declare a new component in an Angular application?

a) angular.json
b) package.json
c) The module file (e.g., app.module.ts)
d) The root HTML file

10. What is the correct syntax for event binding in a component’s template?

a) (event)="handler()"
b) [event]="handler()"
c) {{ event }}
d) handler(event)

11. Which property is used to link an external HTML file to a component?

a) htmlUrl
b) templateUrl
c) viewUrl
d) styleUrl

12. Which command would you use to generate a component named dashboard?

a) ng g c dashboard
b) ng new dashboard
c) ng create dashboard
d) ng add dashboard

13. What is the purpose of the .component.spec.ts file?

a) Defines the component's styles
b) Contains unit tests for the component

c) Stores metadata for the component
d) Manages routing for the component

14. Which lifecycle hook is called when an Angular component is destroyed?

a) ngOnDestroy
b) ngOnInit
c) ngAfterViewInit
d) ngOnChanges

15. How do you pass data from a parent to a child component?

a) Use @Output
b) Use @Input
c) Use @ViewChild
d) Use a service

16. Which lifecycle hook responds to changes in input properties?

a) ngOnInit
b) ngOnChanges
c) ngDoCheck
d) ngAfterContentInit

17. What is an inline template in a component?

a) HTML defined in the templateUrl property
b) HTML written directly in the template property
c) HTML imported from another component
d) HTML defined in the styleUrls property

18. What is the output of the following code snippet?

html
Copy code

```html
<h1>{{ title }}</h1>
```

a) A header with the string title
b) A header displaying the value of the title variable
c) A runtime error
d) Nothing

19. Which command updates the Angular module to declare a new component?

a) ng serve
b) ng generate component
c) ng add module
d) ng build

---

20. What is the benefit of splitting an application into multiple components?

a) Reduces network requests
b) Increases code readability and reusability
c) Speeds up development server build time
d) Prevents style conflicts across components

20 MCQs on Templates and Data Binding

1. What does string interpolation in Angular use?

a) []
b) {}
c) {{ }}
d) ()

2. Which of the following is true about string interpolation?

a) It can modify component data.
b) It binds component data to the template.
c) It is used for listening to events.
d) It sets properties of HTML elements.

3. What is the syntax for property binding?

a) [property]="value"
b) (property)="value"
c) {property: value}
d) {{ property }}

4. Which binding is used to listen for user events?

a) String interpolation
b) Property binding
c) Event binding
d) Two-way binding

5. How do you capture an input event in Angular?

a) {{ input }}
b) (input)="onInput()"
c) [input]="value"
d) [(input)]="value"

6. What is required for two-way binding using [(ngModel)]?

a) ReactiveFormsModule
b) HttpClientModule
c) FormsModule
d) BrowserModule

7. What does two-way binding achieve?

a) One-way communication from component to template.
b) One-way communication from template to component.
c) Synchronization of data between component and template.
d) Dynamic setting of HTML attributes.

8. Which of the following allows dynamic updates of an element's property?

a) Event binding
b) String interpolation
c) Property binding
d) Template reference

9. What is the syntax for two-way data binding?

a) {{ }}
b) [ ]
c) ( )
d) [( )]

10. Which directive is essential for two-way binding in Angular forms?

a) ngModel
b) ngBind
c) ngFor
d) ngIf

11. What will be the result of this template: <h1>{{ 2 + 2 }}</h1>?

a) 2 + 2
b) 4
c) Error
d) Undefined

12. Which event is triggered by clicking a button?

a) (submit)
b) (click)
c) (change)
d) (hover)

13. What is the purpose of [style.color] in Angular?

a) Dynamically bind inline styles to a component.
b) Bind a color to the class attribute.
c) Perform event handling for style changes.
d) Apply global styles to a template.

.

14. How can you pass event data to a handler in Angular?

a) (click)="onClick()"
b) (click)="onClick($event)"
c) (click)="onClick(event)"
d) (click)="event.onClick()"

15. What does [(ngModel)] bind to?

a) HTML attributes
b) Component properties

c) Both the view and the model
d) Event listeners

16. What must be imported to use [ngModel] in an Angular app?

a) ReactiveFormsModule
b) HttpClientModule
c) FormsModule
d) CommonModule

17. How does Angular handle DOM updates with property binding?

a) Manually updates the DOM with JavaScript.
b) Uses an internal rendering engine to sync changes.
c) Triggers a server request for updates.
d) Updates only upon user interaction.

Answer: b) Uses an internal rendering engine to sync changes.

---

18. What does this code do: <input [value]="name">?

a) Binds the input value to the name variable in both directions.
b) Binds the input value to the name variable one-way.
c) Adds a static value of name to the input.
d) Generates a placeholder for name.

19. What happens if you miss importing FormsModule while using [(ngModel)]?

a) The app throws an error.
b) The app ignores the ngModel directive.
c) The app runs, but the binding doesn't work.
d) Nothing happens.

20. Which binding is primarily used for form input fields?

a) Event binding
b) String interpolation
c) Property binding
d) Two-way binding

20 MCQs on Angular Directives

1. What are Angular directives used for?

a) Creating components
b) Manipulating the DOM
c) Handling HTTP requests
d) Defining data models

2. Which of the following is a structural directive?

a) ngClass
b) ngStyle
c) *ngIf
d) @Directive

3. What is the syntax for *ngFor?

a) *ngFor="let item in items"
b) *ngFor="let item of items"
c) *ngFor="item of items"
d) *ngFor="item in items"

4. What does *ngIf="isLoggedIn" do when isLoggedIn is false?

a) Hides the element using CSS.
b) Removes the element from the DOM.
c) Adds a hidden attribute to the element.
d) Throws an error.

5. How can you access the index of an item in *ngFor?

a) *ngFor="let item of items; i=index"
b) *ngFor="let item of items; let i=index"
c) *ngFor="let item; i of items"
d) *ngFor="item, i of items"

6. Which directive is used to dynamically set CSS classes?

a) *ngIf
b) ngClass
c) ngStyle
d) *ngFor

7. What is the result of [ngClass]="{'active': true}"?

a) No class is applied.
b) Adds the active class.
c) Removes the active class.
d) Throws an error.

8. Which of the following is true about ngStyle?

a) It modifies the DOM structure.
b) It applies CSS classes dynamically.
c) It sets inline styles dynamically.
d) It hides or shows elements.

9. What does [ngStyle]="{'color': 'red'}" do?

a) Adds a red CSS class.
b) Changes the text color to red.
c) Removes the color attribute.
d) Sets the background color to red.

10. Which directive would you use to conditionally show an element?

a) ngClass
b) *ngIf
c) ngStyle
d) *ngFor

11. Which directive is NOT structural?

a) *ngIf
b) *ngFor
c) *ngSwitch
d) ngClass

12. What is required to use *ngFor?

a) A collection
b) A boolean expression
c) An object
d) A template reference variable

13. What happens when the expression in *ngIf evaluates to false?

a) The element is hidden using CSS.
b) The element is removed from the DOM.
c) The element becomes transparent.
d) The element stays visible.

14. What does [ngClass]="{highlight: true}" do?

a) Adds a highlight style property.
b) Applies the highlight class.
c) Removes the highlight class.
d) Binds the highlight attribute.

15. Which directive sets the style attribute dynamically?

a) ngClass
b) ngStyle
c) *ngFor
d) *ngIf

16. Which directive is best suited for adding multiple items to the DOM?

a) ngStyle
b) *ngIf
c) *ngFor
d) ngClass

17. Can you use *ngIf and *ngFor on the same element?

a) Yes
b) No
c) Only with additional configuration
d) Only in templates

18. How can you apply multiple styles dynamically using ngStyle?

a) Using a CSS file
b) By passing an object with key-value pairs
c) By adding multiple ngStyle bindings
d) By using a template variable

19. Which directive would you use to highlight an active menu item?

a) *ngIf
b) *ngFor
c) ngClass
d) ngStyle

---

20. What happens to an element with [ngStyle]="{'display': 'none'}"?

a) It is removed from the DOM.
b) It is hidden but remains in the DOM.
c) It causes an error.
d) It becomes partially visible.

15 MCQs on Angular Modules

1. What is the purpose of Angular modules?

a) To declare variables
b) To organize an application into cohesive blocks
c) To manage CSS files
d) To bootstrap the browser

Explanation: Angular modules group related components, directives, pipes, and services into cohesive blocks of functionality.

2. Which decorator is used to define a module?

a) @Component
b) @NgModule
c) @Module
d) @Injectable

Explanation: The @NgModule decorator is used to define an Angular module.

3. What is the default name of the root module in Angular?

a) MainModule
b) RootModule
c) AppModule
d) CoreModule

Explanation: The default root module in an Angular application is called AppModule.

4. What does the bootstrap property in @NgModule specify?

a) The feature modules to load
b) The services to initialize
c) The root component to bootstrap the app
d) The list of imports

Explanation: The bootstrap property identifies the root component that Angular should bootstrap during application initialization.

5. Which core module is required for browser-based applications?

a) HttpClientModule
b) BrowserModule
c) FormsModule
d) CommonModule

Explanation: The BrowserModule is essential for any Angular app that runs in a web browser.

6. How do you create a feature module in Angular?

a) Using the @Component decorator
b) Using the ng generate module CLI command
c) By adding it to the main.ts file
d) By creating a service

Explanation: You can create a feature module using the Angular CLI command ng generate module.

7. What is the purpose of the declarations array in @NgModule?

a) To declare the services
b) To declare components, directives, and pipes
c) To import external modules
d) To bootstrap the application

Explanation: The declarations array lists the components, directives, and pipes that belong to the module.

8. Which module should be used for directives like *ngIf and *ngFor?

a) FormsModule
b) HttpClientModule
c) CommonModule
d) ReactiveFormsModule

Explanation: The CommonModule provides commonly used directives like *ngIf and *ngFor.

9. What is lazy loading?

a) Loading all components during app initialization
b) Delaying the load of feature modules until needed
c) Postponing service initialization
d) None of the above

Explanation: Lazy loading defers the loading of feature modules until they are required by the user.

10. How do you implement lazy loading for a feature module?

a) Using the bootstrap property
b) Using loadChildren in the routes configuration
c) Importing the module in AppModule
d) Using declarations in AppModule

Explanation: Lazy loading is implemented using the loadChildren property in the route definitions.

11. What is a shared module used for?

a) To share state across components
b) To declare and export reusable components, directives, and pipes
c) To handle API requests
d) To provide feature-specific functionality

Explanation: A shared module is used to organize and share reusable components, directives, and pipes across other modules.

12. Which of the following should be included in the exports array?

a) Components to be used in other modules
b) Services to be injected
c) Directives not shared
d) Routes

Explanation: Components, directives, and pipes that need to be available to other modules should be included in the exports array.

13. Which property is used to register services in a module?

Explanation: The providers array in a module is used to register services for dependency injection.

a) declarations
b) providers
c) imports
d) bootstrap

14. What is the purpose of the imports array in a module?

a) To specify root components
b) To include required Angular modules
c) To register global services
d) To export components

Explanation: The imports array specifies the modules that the current module depends on.

15. Can a module import itself?

a) Yes
b) No
c) Only feature modules
d) Only shared modules

Explanation: An Angular module cannot import itself to avoid circular dependencies.

20 MCQs on Dependency Injection and Services

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

a) A design pattern for creating components
b) A method for injecting data into a component
c) A design pattern for injecting dependencies into a class
d) A method for managing the application's lifecycle

2. Which decorator is used to make a service injectable in Angular?

a) @Component
b) @Injectable
c) @Service
d) @Inject

Explanation: The @Injectable decorator marks a class as a service that can be injected via DI.

3. Where do you typically provide a service in Angular to make it available globally?

Explanation: This makes the service available as a singleton throughout the app.

a) In the component’s providers array
b) In the root module’s imports array
c) In the @Injectable decorator with providedIn: 'root'
d) In the component's ngOnInit lifecycle hook

4. What does the providedIn: 'root' syntax in the @Injectable decorator do?

a) It makes the service available only in the component where it’s used.
b) It creates a new instance of the service for each component.
c) It makes the service available globally in the application.
d) It makes the service available only in the module.

Explanation: This is the default setting for making a service a singleton across the app.

5. How do you inject a service into a component in Angular?

a) By importing the service in the component
b) By using the @Inject decorator in the component
c) By adding the service to the component's providers array
d) By adding it to the component's constructor

Explanation: Adding a service to the constructor enables DI to provide the instance automatically.

6. What is the default scope of a service provided in the root injector?

a) Singleton across the entire application
b) Singleton within the module
c) Singleton within the component
d) Scoped to the component

Explanation: Services provided in the root injector are shared across all components.

---

7. What is the purpose of the providers array in an Angular module?

a) To specify which components to bootstrap
b) To register services that can be injected into components
c) To declare pipes and directives
d) To configure the app’s routing

Explanation: The providers array specifies the services available to components within the module.

8. Which lifecycle hook is typically used to fetch data in a component using a service?

a) ngOnChanges
b) ngOnInit
c) ngAfterViewInit
d) ngDoCheck

Explanation: The ngOnInit lifecycle hook is used for initialization logic, including fetching data.

9. What is the result of injecting a service in the constructor of a component?

a) The service is automatically created and available for use in the component.
b) The service needs to be instantiated manually.
c) The component creates multiple instances of the service.
d) The service is only available within the constructor function.

Answer: a) The service is automatically created and available for use in the component.
Explanation: Angular's DI system ensures the service is instantiated and ready to use.

---

10. Which Angular service is typically injected to make HTTP requests?

a) HttpClientModule
b) Http
c) HttpClient
d) HttpRequestService

11. Can you inject services into other services in Angular?

a) No, services cannot inject other services.
b) Yes, services can inject other services.
c) Only root services can inject other services.
d) Services can inject services only within the same module.

Explanation: Services can depend on other services, allowing for complex DI hierarchies.

12. What happens if a service is provided in the component's providers array?

a) It creates a new instance of the service for each component.
b) The service is shared across all components.
c) The service is available globally across the application.
d) The service will be injected only into the root module.

Explanation: Providing a service in the component's providers creates a unique instance for that component.

13. Which of the following is true about Angular services?

a) Services are always created in the component's constructor.
b) Services are injected into modules.
c) Services are used to store business logic and can be shared across components.
d) Services cannot be used to manage state in an Angular app.

Explanation: Angular services encapsulate logic and data that can be reused throughout the app.

14. Which of the following is an advanced provider configuration in Angular DI?

a) useClass
b) useFactory
c) useValue
d) All of the above

Explanation: Angular allows useClass, useFactory, and useValue for advanced DI configurations.

Answer: d) All of the above

---

15. What is the role of tokens in Angular's DI system?

a) Tokens are used to identify the services to be injected.
b) Tokens store the values returned by services.
c) Tokens are used to identify components.
d) Tokens create instances of services.

16. Can you use DI in Angular with non-class values such as strings or objects?

a) No, DI can only be used with classes.
b) Yes, DI can be used with strings or objects using tokens.
c) Yes, but only with global variables.
d) No, DI is strictly for components only.

17. What is the advantage of using DI in Angular applications?

a) It reduces the complexity of the application
b) It makes it easier to manage the lifecycle of components
c) It allows for better testing and decoupling of components and services
d) It allows services to be reused only within a module

Explanation: DI simplifies testing and promotes separation of concerns.

18. Which service can be used to mock the behavior of real services in Angular unit tests?

a) HttpClient
b) TestBed
c) MockDataService
d) MockHttpClient

Explanation: TestBed allows mocking and configuring dependencies during testing.

19. How can Angular's DI system benefit testing?

a) It allows services to be easily replaced with mock services.
b) It ensures components are tightly coupled with services.

Explanation: DI enables the injection of mock implementations for testing.

c) It eliminates the need for testing components.
d) It makes services unavailable in the test environment.

---

20. How does Angular’s DI improve the maintainability of an application?

a) By making services more difficult to test
b) By forcing components to create their own services
c) By reducing the amount of manual instantiation and ensuring that components have access to the right services
d) By requiring developers to manually manage the lifecycle of services

20 MCQs on Angular Routing

1. What is the purpose of the Angular RouterModule?

a) To configure and manage routes
b) To define components in the application
c) To handle HTTP requests
d) To store routing data

Explanation: RouterModule is used to configure the navigation paths and manage routing in Angular applications.

2. What is the first step in setting up routing in an Angular application?

a) Import the RouterModule and Routes
b) Add <router-outlet> to the template
c) Define routes in the component
d) Use the forRoot() method in AppModule

Explanation: You need to import RouterModule and define Routes to configure the routing for your application.

---

3. What is the role of the <router-outlet> directive in Angular?

a) It defines the navigation paths
b) It holds the content of the current route
c) It manages route parameters
d) It defines the navigation structure

Explanation: <router-outlet> is a placeholder where the component for the current route is displayed.

4. How do you define a route in Angular?

a) By creating a service for each path
b) By using the RouterModule's routes configuration

b) By using the RouterModule's routes configuration

Explanation: Routes are defined in the RouterModule’s configuration to specify path-to-component mapping.

c) By binding the path to a component in the AppModule
d) By using the ngFor directive

5. What is the method used to configure root routes in Angular?

a) forChild()
b) forRoot()
c) initialize()
d) configure()

Explanation: forRoot() is used to configure the root routes in the main application module (AppModule).

6. What is the difference between forRoot() and forChild()?

a) forRoot() is used in child modules, forChild() in root modules
b) forRoot() is used in the root module, forChild() in feature modules
c) Both are used in the root module
d) There is no difference

b) forRoot() is used in the root module, forChild() in feature modules

---

7. Which of the following is used to navigate between routes programmatically in Angular?

a) routerNavigate()
b) navigateByUrl()
c) Router.navigate()
d) navigateTo()

Explanation: Router.navigate() is used to programmatically navigate between routes.

8. What type of value does a route path support?

a) Static values only
b) Dynamic values only
c) Static and dynamic values
d) Only numeric values

Explanation: Route paths can include static values (e.g., /home) and dynamic parameters (e.g., /user/:id).

---

9. How can you pass parameters to a route in Angular?

a) Using queryParams
b) Using URL parameters (e.g., :id)
c) Using the ActivatedRoute service
d) All of the above

Explanation: Parameters can be passed via query parameters, URL parameters, or the ActivatedRoute service.

10. What is the directive used to highlight an active link in Angular?

a) routerActive
b) routerLinkActive
c) activeClass
d) activeLink

b) routerLinkActive

Explanation: routerLinkActive is used to apply a CSS class to the active route link.

11. What happens when a route is activated in Angular?

a) The URL is updated
b) The corresponding component is displayed
c) The browser reloads
d) Both a and b

d) Both a and b

Explanation: When a route is activated, both the URL is updated and the corresponding component is displayed.

12. Which directive is used to create navigation links between routes?

a) routerNavigate
b) routerLink
c) navigateTo
d) routeLink

b) routerLink

Explanation: routerLink is used to create links between routes in Angular applications.

13. What is the use of lazy loading in Angular routing?

a) It improves performance by loading feature modules only when needed
b) It disables the route navigation
c) It loads all components at once
d) It disables route parameters

a) It improves performance by loading feature modules only when needed

Explanation: Lazy loading delays loading of feature modules until they are required, improving app performance.

---

14. Which method can be used to handle nested routes?

a) navigateByUrl()
b) Using child routes in the RouterModule configuration
c) forChild() method
d) navigate()

b) Using child routes in the RouterModule configuration

15. What is route guard in Angular?

a) A service that protects routes from being accessed
b) A directive for managing route transitions
c) A feature for loading routes lazily
d) A mechanism for customizing URL paths

a) A service that protects routes from being accessed

Explanation: Route guards are used to prevent access to routes based on conditions like authentication or permissions.

16. How can you handle 404 errors (page not found) in Angular routing?

a) By adding a wildcard route (**)
b) By using the catchError operator
c) By using a custom error handler
d) By using route guards

a) By adding a wildcard route ()**

Explanation: A wildcard route (**) is used to catch any undefined routes and display a 404 page or redirect.

17. What does the ActivatedRoute service provide in Angular?

a) Access to the current route's information, such as parameters and query params
b) Methods for navigating between routes
c) The ability to update the URL manually
d) Access to the component's lifecycle hooks

: a) Access to the current route's information, such as parameters and query params

18. How can you apply a CSS class to the active route link in Angular?

a) Use the activeLink directive
b) Use routerLinkActive directive
c) Use the activeRoute directive
d) Apply it manually in the component

b) Use routerLinkActive directive

Explanation: routerLinkActive applies a CSS class to the active link in the router navigation.

19. What is the purpose of using pathMatch: 'full' in routing?

a) It ensures the exact path is matched
b) It allows partial matches of the path
c) It disables route activation
d) It restricts the path to specific components

a) It ensures the exact path is matched

Explanation: pathMatch: 'full' ensures that the route is activated only when the entire URL path matches.

20. What is the best practice for organizing routes in large Angular applications?

a) Define all routes in a single file
b) Use child modules and lazy loading
c) Use only static routes
d) Avoid using route parameters

b) Use child modules and lazy loading

Explanation: To keep the application scalable and maintainable, large applications should use lazy-loaded feature modules and nested routing.

20 MCQs on Angular Template-driven Forms

1. What is the main module required to use template-driven forms in Angular?

a) FormsModule
b) ReactiveFormsModule
c) RouterModule
d) HttpClientModule

a) FormsModule

Explanation: FormsModule is the main module required to use template-driven forms in Angular.

2. Which directive is used to create two-way data binding for form controls in Angular?

a) ngModel
b) formControlName
c) ngIf
d) ngFor

a) ngModel

Explanation: The ngModel directive is used to create two-way data binding for form controls.

3. What does the required attribute do in Angular template-driven forms?

a) It ensures the field is required for the form to be valid.
b) It makes the field read-only.
c) It triggers form submission.
d) It validates the field against a regular expression.

a) It ensures the field is required for the form to be valid.

Explanation: The required attribute in a template-driven form makes the field mandatory for form submission.

4. Which of the following is used to validate email input in template-driven forms?

a) pattern
b) email
c) required
d) minlength

b) email

Explanation: The email validator is used to validate email input in template-driven forms.

5. What is the purpose of ngForm in Angular?

a) It creates a new form control in the component class.
b) It binds form controls to component properties.
c) It is used to reference the form in the template.
d) It performs form submission automatically.

: c) It is used to reference the form in the template.

Explanation: ngForm is a directive that provides a way to reference the form object in the template.

6. How can form validation feedback be shown to users in Angular template-driven forms?

a) By using the *ngFor directive
b) By using *ngIf to display error messages
c) By using the ngClass directive
d) By binding error messages to form control properties

b) *By using ngIf to display error messages

Explanation: Validation error messages are typically displayed conditionally using *ngIf based on the form control's state.

7. How can you disable the submit button in Angular template-driven forms until the form is valid?

a) Use [disabled]="!form.valid"
b) Use [disabled]="form.invalid"

b) Use [disabled]="form.invalid"

Explanation: You can disable the submit button by checking the form’s validity with form.invalid.

c) Use [disabled]="!form.submitted"
d) Use [disabled]="form.invalid || !form.submitted"

8. Which of the following is a built-in validator in Angular template-driven forms?

a) required
b) maxLength
c) positiveNumber
d) customValidator

a) required

Explanation: required is a built-in validator in Angular template-driven forms.

9. What is the correct syntax for binding a form control to the component in a template-driven form?

a) [ngModel]="value"
b) [(ngModel)]="value"
c) ngModel="value"
d) [(value)]="ngModel"

b) [(ngModel)]="value"

Explanation: [(ngModel)]="value" is the syntax for two-way data binding in template-driven forms.

---

10. Which of the following Angular directives is used to handle form submission?

a) ngSubmit
b) ngModel
c) formControlName
d) ngFor

a) ngSubmit

Explanation: ngSubmit is the directive used to handle form submission in template-driven forms.

11. What does the email validator do in template-driven forms?

a) It ensures the input value is a valid email address.
b) It ensures the input is only alphabetic.
c) It ensures the input has at least one symbol.
d) It ensures the input is not empty.

a) It ensures the input value is a valid email address.

Explanation: The email validator checks whether the input matches the pattern for a valid email address.

12. How can you make a field optional in a template-driven form?

a) Use required="false"
b) Do not use any validators
c) Use optional="true"
d) Use ngModel without any validation

b) Do not use any validators

Explanation: If no validators are applied, the field is optional.

13. How can custom form validation be added to a template-driven form in Angular?

a) By creating a custom validator function and using ngModel
b) By using the pattern attribute
c) By using ngSubmit
d) By creating a FormControl object

a) By creating a custom validator function and using ngModel

Explanation: You can create a custom validator function and apply it via ngModel in template-driven forms.

14. How do you access the form values in the component after submission?

a) form.getValues()
b) form.get()
c) form.value
d) form.controls

c) form.value

Explanation: The form.value property provides access to the current values of all form controls.

15. How can you prevent form submission if it's invalid?

a) By using [disabled]="!form.valid" on the submit button
b) By using form.preventSubmit()
c) By disabling the form controls
d) By using ngModel with submit event

a) By using [disabled]="!form.valid" on the submit button

Explanation: Disabling the submit button when the form is invalid prevents form submission.

16. How do you display validation errors after a form control is touched?

a) By using touched property of the control
b) By using ngSubmit event
c) By using ngModel validation only
d) By using form.get('controlName').valid

a) By using touched property of the control

Explanation: The touched property of a form control is used to check if the control has been interacted with and can trigger validation feedback.

17. How do you handle multiple forms in a single component?

a) Use ngFor to loop through forms
b) Use different ngForm references for each form
c) Use multiple ngModel references
d) Use the ngSubmit directive

b) Use different ngForm references for each form

Explanation: You can handle multiple forms by using different ngForm references for each form.

18. Which of the following is a way to handle form submission in Angular template-driven forms?

a) Using ngSubmit on the <form> element
b) Using ngClick on the submit button
c) Using formSubmit method in the component
d) Using ngFormSubmit directive

a) Using ngSubmit on the <form> element

Explanation: ngSubmit is used to handle form submission in template-driven forms.

19. Which of the following is used to apply validation feedback based on form control status?

a) ngModelStatus
b) ngIf
c) ngSubmit
d) ngControl

b) ngIf

Explanation: ngIf is often used to conditionally display error messages based on the validation status of a form control.

---

20. What is the purpose of the minlength validator in Angular?

a) It ensures the input is a valid email.
b) It sets a minimum length for a text input.
c) It ensures a form is not empty.
d) It checks if the input matches a specific pattern.

b) It sets a minimum length for a text input.

Explanation: The minlength validator ensures that the input value meets the specified minimum length requirement.

20 Multiple Choice Questions (MCQs) on Angular Pipes

1. Which of the following is a built-in pipe in Angular?

a) ReversePipe
b) DatePipe
c) CustomPipe
d) FormatPipe

b) DatePipe

Explanation: DatePipe is a built-in Angular pipe that formats dates.

2. How do you format a date in Angular using the DatePipe?

a) {{ today | date:'dd/MM/yyyy' }}
b) {{ today | date }}
c) {{ today | date:'yyyy-MM-dd' }}
d) All of the above

d) All of the above

Explanation: You can format a date using {{ today | date:'dd/MM/yyyy' }}, {{ today | date }}, or {{ today | date:'yyyy-MM-dd' }} in Angular.

3. What does the UppercasePipe do?

a) Converts text to lowercase
b) Converts text to uppercase
c) Formats dates
d) Converts numbers to currency

b) Converts text to uppercase

Explanation: The UppercasePipe transforms text to uppercase.

4. How do you format a number as currency using the CurrencyPipe?

a) {{ 1000 | currency }}
b) {{ 1000 | currency:'USD' }}
c) {{ 1000 | currency:'USD':true }}
d) All of the above

d) All of the above

Explanation: The CurrencyPipe can format numbers as currency, with optional parameters such as a specific currency code (e.g., USD).

5. Which pipe would you use to format a number into a percentage?

a) DatePipe
b) PercentPipe
c) UppercasePipe
d) CurrencyPipe

b) PercentPipe

Explanation: The PercentPipe is used to format numbers as percentages.

6. What does the DecimalPipe do?

a) Formats a string into a decimal
b) Formats a number with specified decimal places
c) Converts a string into a currency
d) Converts a date into a string

b) Formats a number with specified decimal places

Explanation: The DecimalPipe formats a number with the desired number of decimal places.

---

7. How do you display an object in JSON format using Angular?

a) {{ object | json }}
b) {{ object | stringify }}
c) {{ object | json.stringify }}
d) {{ object | format:'json' }}

a) {{ object | json }}

Explanation: The JsonPipe is used to display an object in JSON format.

8. Which of the following pipes can be used to transform the text in both uppercase and lowercase?

a) UppercasePipe and LowercasePipe
b) CurrencyPipe and UppercasePipe
c) UppercasePipe and DecimalPipe
d) JsonPipe and UppercasePipe

a) UppercasePipe and LowercasePipe

Explanation: The UppercasePipe and LowercasePipe can transform text to uppercase and lowercase, respectively.

9. What is the primary function of the AsyncPipe?

a) Converts a number to currency
b) Subscribes to an Observable and returns the latest value
c) Transforms a string into a JSON format
d) Formats a date

b) Subscribes to an Observable and returns the latest value

Explanation: The AsyncPipe subscribes to an Observable and returns its latest emitted value.

---

10. How do you create a custom pipe in Angular?

a) By extending the Pipe class
b) By creating a class with the Pipe decorator and implementing PipeTransform

c) By using @Injectable()
d) By creating a directive

b) By creating a class with the Pipe decorator and implementing PipeTransform

Explanation: A custom pipe is created by defining a class with the @Pipe decorator and implementing the PipeTransform interface.

11. Which decorator is used to create a custom pipe in Angular?

a) @Injectable()
b) @Pipe()
c) @Directive()
d) @Component()

b) @Pipe()

Explanation: The @Pipe decorator is used to create a custom pipe.

12. What method must be implemented in a custom pipe?

a) transform()
b) ngOnInit()
c) ngOnChanges()
d) process()

a) transform()

Explanation: The transform() method must be implemented in a custom pipe to define the transformation logic.

13. Which of the following is true about custom pipes?

a) They can only be used inside components
b) They can be used to format data
c) They can only transform numbers
d) They are similar to Angular services

b) They can be used to format data

Explanation: Custom pipes can be used to transform or format data based on specific logic.

14. How do you apply a custom pipe in a template?

a) By using the pipe's name in the template, like {{ value | reverse }}
b) By injecting the pipe into the component
c) By declaring the pipe in the module's imports
d) By using the transform() function directly

a) By using the pipe's name in the template, like {{ value | reverse }}

Explanation: You apply a custom pipe in the template by using its name followed by a pipe (|), such as {{ value | reverse }}.

15. Can a custom pipe be created without any input parameters?

a) Yes, a pipe can have no input parameters
b) No, pipes must always have input parameters
c) It depends on the Angular version
d) A pipe must always be a function

a) Yes, a pipe can have no input parameters

Explanation: Custom pipes can be created with or without input parameters.

---

16. What is the purpose of the pure option in Angular pipes?

a) Makes the pipe reusable
b) Makes the pipe stateless
c) Ensures the pipe runs only when the input changes
d) Forces the pipe to run every time

c) Ensures the pipe runs only when the input changes

Explanation: The pure option ensures the pipe is only re-evaluated when the input changes. By default, pipes are pure, but you can make them impure if necessary.

7. Which of the following is a valid custom pipe implementation?

a) transform(value: string): number
b) transform(value: any): any
c) pipe(value: any): any
d) toString(value: any): string

b) transform(value: any): any

Explanation: A valid custom pipe should implement the transform() method, which takes a value and returns the transformed value.

18. Which of the following pipes is not a built-in pipe in Angular?

a) JsonPipe
b) UppercasePipe
c) ReversePipe
d) DatePipe

c) ReversePipe

Explanation: ReversePipe is not a built-in pipe in Angular, though it can be custom-created.

19. Which Angular pipe is most commonly used for localization (i18n)?

a) CurrencyPipe
b) I18nSelectPipe
c) PercentPipe
d) DatePipe

d) DatePipe

Explanation: The DatePipe is commonly used for formatting dates according to the locale in Angular applications for internationalization (i18n).

20. Which of the following is a use case for the AsyncPipe?

a) To format strings in uppercase
b) To subscribe to an observable and display the latest value
c) To convert numbers into percentages
d) To format a date based on locale

b) To subscribe to an observable and display the latest value

Explanation: The AsyncPipe is used to subscribe to an observable and automatically update the view with the latest value.