Interview Question & Answers

1. Why were client-side frameworks like Angular introduced?

- **To simplify front-end development:** Before frameworks, building complex web applications involved a lot of manual coding for tasks like handling user interactions, updating the page without reloading, and managing data.
- **To improve efficiency and maintainability:** Frameworks provide reusable components, standardized structures, and tools that make development faster, easier to maintain, and less prone to errors.

2. What are some of the advantages of Angular over other frameworks?

- **TypeScript Support:** Angular leverages TypeScript, a superset of JavaScript, which enhances code maintainability, readability, and reduces errors through static type checking.
- **Component-Based Architecture:** Angular promotes a modular approach with components, making it easier to build, test, and reuse UI elements.
- **Two-Way Data Binding:** This feature simplifies data synchronization between the model and the view, reducing boilerplate code and improving developer productivity.
- **Cross-Platform Development:** Angular can be used to build web, mobile, and desktop applications using technologies like Progressive Web Apps (PWAs) and NativeScript.

3. What are the advantages of Angular over React?

Angular	**React**
Angular supports bidirectional data binding as well as mutable data.	React only supports unidirectional and immutable data binding.
The biggest benefit of Angular is that it enables dependency injection.	React allows us to either accomplish it ourselves or with the aid of a third-party library.
Angular can be used in both mobile and web development.	React can only be used in UI development only.
Angular features a wide wide range of tools, libraries, frameworks, plugins, and so on that make development faster and more fun.	In React we can use third-party libraries for any features.
Angular uses Typescript.	React uses Javascript.

4. List out differences between AngularJS and Angular?

Features	AngularJS	Angular
Architecture	AngularJS uses MVC or Model-View-Controller architecture, where the Model contains the business logic, the Controller processes information and the View shows the information present in the Model.	Angular replaces controllers with Components. Components are nothing but directives with a predefined template.
Language	AngularJS uses JavaScript language, which is a dynamically typed language.	Angular uses TypeScript language, which is a statically typed language and is a superset of JavaScript. By using statically typed language, Angular provides better performance while developing larger applications.
Mobile Support	AngularJS does not provide mobile support.	Angular is supported by all popular mobile browsers.
Structure	While developing larger applications, the process of maintaining code becomes tedious in the case of AngularJS.	In the case of Angular, it is easier to maintain code for larger applications as it provides a better structure.
Expression Syntax	While developing an AngularJS application, a developer needs to remember the correct ng-directive for binding an event or a property.	Whereas in Angular, property binding is done using "[ ]" attribute and event binding is done using "( )" attribute.

5. How are Angular expressions different from JavaScript expressions?

- **Context:**
- **Angular Expressions:** Evaluated within the **local scope** of the Angular component or controller. This means they have access to the component's properties, methods, and any other variables defined within that specific component's context.
- **JavaScript Expressions:** Evaluated within the **global scope** (usually the window object in a browser). They have access to globally defined variables and functions.
- **Purpose:**
- **Angular Expressions:** Primarily designed for **data binding** within Angular templates. They allow you to display component data, bind to user input, and control the appearance of the UI based on data changes.
- **JavaScript Expressions:** Have a much broader purpose, used for various tasks like:
    - **Calculations:** let result = num1 + num2;

    - **Assignments:** myVariable = "some value";
    - **Conditional checks:** if (condition) { ... }
    - **Function calls:** myFunction();
- **Syntax and Features:**
- **Angular Expressions:**
    - **Limited syntax:** Primarily for data binding and simple logic.
    - **Built-in filters:** For formatting data (e.g., currency, date, uppercase).
    - **One-time binding:** For performance optimization.
- **JavaScript Expressions:**
    - **Full JavaScript syntax:** Access to all JavaScript features (control flow, functions, objects, etc.).
    - **No built-in filters:** Requires custom functions for data formatting.

6. Single Page Applications (SPAs)

- **Definition:** SPAs are web applications that load a single HTML page and dynamically update that single page with the content the user requests.
- **Key Characteristics:**
    - **Dynamic Updates:** Instead of loading entire new pages, SPAs use JavaScript to update the content within the existing page. This creates a more fluid and responsive user experience.
    - **Faster Interactions:** Since only the necessary data is transferred, SPAs can be much faster than traditional multi-page applications, especially on slower connections.
    - **Improved User Experience:** The smooth transitions and lack of page reloads enhance the overall user experience.
- **Examples:** Gmail, Google Maps, Facebook

7. Templates in Angular

- **Definition:** Templates are the building blocks of the user interface in Angular. They define the HTML structure of a component's view.
- **Key Features:**
    - **Data Binding:** Templates use special syntax (like interpolation {{ }} or property binding [property]="expression") to bind data from the component's class to the HTML.
    - **Directives:** Templates can incorporate directives, which are special instructions that modify the behavior of DOM elements.
    - **Components:** Each component has its own associated template that defines how it should be rendered.

8. Directives in Angular

- **Definition:** Directives are special classes that add new behavior to existing DOM elements. They act as instructions to Angular on how to transform the DOM.
- **Types:**
    - **Structural Directives:** These directives change the DOM structure itself. Examples include:
        - *ngIf: Conditionally shows or hides elements.
        - *ngFor: Repeats a section of the template based on an array.
    - **Attribute Directives:** These directives change the appearance or behavior of an existing DOM element. Examples include:
        - ngClass: Dynamically adds or removes CSS classes.
        - ngStyle: Dynamically sets inline styles.

9. Components, Modules, and Services in Angular

- **Components:**
    - **Definition:** The fundamental building blocks of an Angular application. Each component encapsulates a portion of the UI (its template and its logic).
    - **Responsibilities:**
        - Defines the UI structure (template).
        - Handles user interactions (events).
        - Manages the data specific to that part of the UI.
- **Modules:**
    - **Definition:** A container for a cohesive block of functionality related to an application domain, a workflow, or a feature area.
    - **Responsibilities:**
        - Organize and group related components, directives, and services.
        - Control what parts of the application are visible to other parts.
        - Lazy loading: Load modules only when they are needed, improving initial load time.
- **Services:**
    - **Definition:** Reusable blocks of code that provide specific functionality, such as:
        - Fetching data from a server (HTTP requests).
        - Logging events.
        - Working with local storage.
    - **Key Advantages:**
        - **Reusability:** Services can be injected into multiple components, making your code more modular and easier to maintain.
        - **Testability:** Services are easily testable in isolation.

- **Improved Code Organization:** Separates business logic from component logic.

10. What is the scope?

In Angular, a scope is an object that refers to the application model. It is a context in which expressions can be executed. These scopes are grouped hierarchically, comparable to the DOM structure of the application. A scope aids in the propagation of various events and the monitoring of expressions.

11. Data Binding in Angular

- **Data binding** is the automatic synchronization of data between the component's class and its template. This means changes in the component's properties are automatically reflected in the view, and vice versa.

12. Two-Way Data Binding

- **Two-way data binding** is a special type of data binding where changes made in the view are immediately reflected in the component's properties, and vice versa. It's achieved using the [(ngModel)] directive.

13. Decorators and their Types in Angular

- **Decorators** are special expressions that add metadata to a class, method, or property. They are prefixed with the @ symbol.
- **Common types of decorators in Angular:**
  - **@Component:** Defines a component, including its template, styles, and metadata.
  - **@Directive:** Creates a custom directive that can be used to manipulate the DOM.
  - **@Injectable:** Marks a class as a service that can be injected into other components.
  - **@Input:** Defines an input property for a component, allowing data to be passed from a parent component.
  - **@Output:** Defines an output property for a component, allowing it to emit events to parent components.

14. Annotations in Angular

- **Annotations** are similar to decorators in that they provide metadata about a class, method, or property. However, they are a more general term and can be used in various contexts, not just Angular.

15. Pure Pipes

- **Pure pipes** are optimized for performance. They are only executed when their input arguments change. This means if the input arguments remain the same, the pipe will not be re-executed, improving application performance.

16. Impure Pipes

- **Impure pipes** are executed every time the Angular change detection cycle runs. This is useful for cases where the pipe needs to be re-executed even if the input arguments haven't changed, such as when dealing with asynchronous data or user interactions.

17. Pipe Transform Interface

- The PipeTransform interface defines the contract that all pipes must adhere to. It has a single method, transform(), which takes the input value and any optional arguments and returns the transformed value.

18. Sharing Data from Parent to Child Component

You have to share the data amongst the components in numerous situations. It may consist of unrelated, parent-child, or child-parent components.

The @Input decorator allows any data to be sent from parent to child.

```
// Parent Component
@Component({
  // ...
})
export class ParentComponent {
  parentData = 'Data from Parent';
}

// Child Component
@Component({
  // ...
})
export class ChildComponent {
  @Input() childData: string;
}
```

19. TypeScript Class with Constructor and Function

```
class MyClass {
  constructor(public name: string) {}
```

```
  greet() {
    console.log(`Hello, ${this.name}!`);
  }
}
```

20. Explain MVVM architecture

- **Model-View-ViewModel** is a design pattern that separates the user interface (View) from the business logic (Model) using an intermediary (ViewModel).
- **Key benefits:**
  - **Improved testability:** The ViewModel can be easily tested independently of the View.
  - **Better maintainability:** Changes to the UI or business logic can be made with minimal impact on other parts of the application.
  - **Enhanced developer productivity:** The separation of concerns allows for better code organization and easier collaboration.

21. What is a bootstrapping module?

- **Definition:** The bootstrapping module is the root module of an Angular application. It's typically named AppModule and is responsible for:
  - Declaring the root component of the application.
  - Importing other necessary modules (e.g., BrowserModule, FormsModule, HttpClientModule).
  - Providing services that need to be available throughout the application.

22. What is Change Detection, and how does the Change Detection Mechanism work?

- **Definition:** Change detection is the process of determining if the component's data has changed and updating the view accordingly.
- **Mechanism:** Angular uses a change detection mechanism to efficiently update the UI only when necessary.
  - **Default Strategy:** Change detection checks for changes in all components in the application tree.
  - **OnPush Strategy:** More efficient, it only checks for changes if:
    - The component's input properties change.
    - An observable stream emits a new value.
    - The component explicitly marks itself as dirty.

23. What is AOT compilation? What are the advantages of AOT?

- **Definition:** AOT compilation translates Angular templates into highly optimized JavaScript code during the build process.

- **Advantages:**
    - **Faster Rendering:** The browser doesn't need to spend time compiling templates at runtime, leading to faster initial load times.
    - **Improved Security:** Potential security risks like template injection are mitigated during the build process.
    - **Smaller Bundle Size:** AOT compilation can result in smaller application bundles, improving performance.

24. What are HTTP interceptors ?

- **Definition:** HTTP interceptors are a mechanism to intercept and modify HTTP requests and responses in Angular. They can be used for:
    - Adding headers to requests (e.g., authentication tokens).
    - Logging requests and responses.
    - Handling errors globally.
    - Transforming request or response data.

25. What is transpiling in Angular ?

- **Definition:** Transpiling is the process of converting TypeScript code (which includes features like classes, interfaces, and decorators) into plain JavaScript that can be understood and executed by the browser.

26. What is ngOnInit?

- **Definition:** ngOnInit is a lifecycle hook that is called after Angular has initialized the component's data-bound properties. It's a common place to:
    - Perform initial data loading or setup tasks.
    - Subscribe to observables.

27. What does Angular Material means?

- **Definition:** Angular Material is a UI component library that provides a set of reusable UI components (e.g., buttons, cards, dialogs, tables) that adhere to Google's Material Design guidelines. It helps build visually appealing and consistent user interfaces.

28. What exactly is the router state?

- **Definition:** Router state represents the current URL, the active route, and the route parameters. It provides information about the current navigation state of the application.

29. What are router links?

- **Definition:** Router links are used to navigate between different routes within an Angular application. They are typically used within <a> tags or other navigation elements.

30. What are lifecycle hooks in Angular? Explain a few lifecycle hooks?

Definition: Lifecycle hooks are methods that are called at specific points in the lifecycle of a component. They allow you to perform actions at different stages of the component's existence.

- **Examples:**
  - ngOnInit: Called after the component's data-bound properties are initialized.
  - ngOnChanges: Called when input properties of the component change.
  - ngOnDestroy: Called before the component is destroyed.

31. What is the Component Decorator in Angular?

Definition: The @Component decorator acts as a blueprint for Angular to understand and create a component instance. It's where you provide crucial metadata that defines the component's behavior and structure.

- **Key Properties:**
  - **selector**: Defines the CSS selector that will be used to match this component in the template.
  - **template**: Defines the HTML template for the component (can be inline or external).
  - **templateUrl**: Specifies the path to an external HTML file for the component's template.
  - **styleUrls**: Specifies the path to CSS or SCSS files for the component's styles.
  - **providers**: An array of services that should be provided to this component and its children.

32. What are property decorators?

- **Definition:** Property decorators modify the behavior of a class property.
- **Examples:**
  - **@Input():** Defines an input property that allows data to be passed from a parent component to this component.
  - **@Output():** Defines an output property that allows this component to emit events to its parent.
  - **@HostBinding():** Binds a property to a host element (the element that the component is attached to).

33. What are Method decorators?

- **Definition:** Method decorators modify the behavior of a class method.
- **Example:**
    - **@HostListener():** Listens for events on the host element and executes a method in response.

34. What are class decorators?

Definition: Class decorators modify the behavior of an entire class.

- **Example:**
    - **@Injectable():** Marks a class as a service that can be injected into other components.

35. What exactly is a parameterized pipe?

- **Definition:** Parameterized pipes allow you to pass additional arguments to a pipe. This provides more flexibility and customization.
- **Example:**
    - {{ myNumber | number:'1.2-3' }} - Formats a number with 1 digit before the decimal point and 3 digits after.

36. What are pipes in Angular explain with an example?

- **Definition:** Pipes are pure functions that transform data before it's displayed in the template.
- **Examples:**
    - **DatePipe**: Formats dates.
    - **CurrencyPipe**: Formats currency values.
    - **UpperCasePipe**: Converts text to uppercase.
    - **Custom Pipes:** You can create your own custom pipes to perform specific data transformations.

37. Explain the concept of Dependency Injection?

- **Definition:** Dependency Injection is a design pattern where a class receives its dependencies (services, other classes) as parameters in its constructor.
- **Benefits:**
    - **Improved Testability:** Makes it easier to test components in isolation.
    - **Loose Coupling:** Reduces dependencies between components.
    - **Better Maintainability:** Makes it easier to modify or replace dependencies.

38. How are observables different from promises?

- **Observables**:
    - Can emit multiple values over time.

- Support cancellation (you can unsubscribe from an observable).
- Provide a rich set of operators for manipulating data streams (map, filter, reduce, etc.).
- Well-suited for asynchronous operations like network requests, user input events, and timers.

➢ **Promises**:

- Can only emit a single value (either a resolved value or an error).
- Cannot be canceled once initiated.
- Have fewer built-in operators for data manipulation.
- Best suited for single-value asynchronous operations.

39. Explain string interpolation and property binding in Angular?

- **String Interpolation:**
  - Displays the value of a component property within the template.
  - Syntax: {{ property }}
- **Property Binding:**
  - Binds a component property to an HTML attribute.
  - Syntax: [property]="expression"

40. What are RxJs in Angular?

Definition: RxJS (Reactive Extensions for JavaScript) is a library for working with asynchronous data streams. RxJS is widely used for handling HTTP requests, managing user interactions, and building reactive applications.

- **Key Concepts:**
  - **Observables:** Represent streams of data that can emit multiple values over time.
  - **Operators:** Functions that transform or combine observables.
  - **Subjects:** Special types of observables that can both emit values and subscribe to other observables.

41. What is view encapsulation in Angular?

- **Definition:** In Angular, **view encapsulation** is a mechanism that controls how styles defined within a component's stylesheet interact with other parts of the application. It helps to prevent style conflicts and maintain component isolation.
- **Types:**
  - **Emulated:** This is the default encapsulation strategy. Styles defined within the component's stylesheet are scoped to elements with these unique attributes. This provides a good balance between encapsulation and performance.
  - **Shadow DOM:** This strategy creates a separate DOM tree for the component's template, isolating it from the rest of the application's DOM.

Provides the strongest encapsulation, isolating styles completely within the component.
    - **None:** Styles are global and can affect other components. This can lead to style conflicts and make it difficult to maintain component isolation.

42. What is Eager and Lazy loading?

- **Eager Loading:** All modules and their dependencies are loaded when the application starts. This can lead to larger initial bundle sizes.
- **Lazy Loading:** Modules are loaded only when they are needed, improving initial load times and reducing memory usage.

43. Angular by default, uses client-side rendering for its applications?

- **Yes:** By default, Angular applications use client-side rendering. This means that the browser downloads the application's code and renders the UI in the user's browser.

44. What happens when you use the script tag within a template?

- Angular detects the value as unsafe and sanitizes it automatically by removing script tag inside your Angular templates and keep your application safe.
- **Security:** Angular considers it a security risk (like a potential attack).
- **Removal:** Angular removes the <script> tag to prevent malicious code from running.
- **Warning:** You'll see a warning in the browser console.
- **Alternatives:**
    - Use Angular's built-in mechanisms like Data Binding, Directives ext…
    - Consider using a library like @nguniversal for server-side rendering.

45. How can I include SASS into an Angular project?

- If You may use the ng new command while building your project using angular CLI. All of your components are generated using preset sass files.
    - ng new <Your_Project_Name> --style = sass
- If you want to change the style of your project, use the ng set command.
    - ng set defaults.styleExt scss

46. How do you deal with errors in observables?

- Instead of depending on try/catch, which is useless in an asynchronous context, you may manage problems by setting an error callback on the observer.
- You may create an error callback as shown below.

```
myObservable.subscribe({
  next(number) { console.log('Next number: ' + number)},
  error(err) { console.log('An error received ' + err)}
});
```

47. How does one share data between components in Angular?

- **@Input() and @Output():** For parent-to-child and child-to-parent communication.
- **Services:** Create a shared service to store and share data across multiple components.
- **State Management Libraries:** Libraries like NgRx or Akita provide more advanced state management solutions.

48. How do you choose an element from a component template?

- **ViewChildren:** Retrieve a list of child components or elements within the template.
- **ViewChild:** Retrieve a single child component or element.
- **ElementRef:** Access the underlying native element of a component.

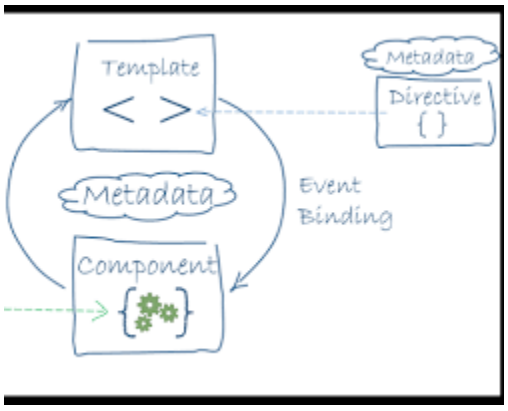
49. What is Angular Framework?

- **Angular** is a TypeScript-based, open-source front-end web application framework led by the Angular Team at Google and a community of individuals and corporations.
- It's used to build dynamic web applications.
- It's a complete rewrite from the same team that built AngularJS.

50. What is TypeScript?

- **TypeScript** is a superset of JavaScript that adds static typing to the language.
- It's developed and maintained by Microsoft.
- TypeScript code is eventually compiled into plain JavaScript, making it compatible with any browser or JavaScript engine.

51. Pictorial Diagram of Angular Architecture



52. Key Components of Angular

- **Modules:** A container for a cohesive block of functionality.
- **Components:** Building blocks of the UI, each with its own template and logic.

- **Templates:** HTML that forms the view, often with special syntax for data binding.
- **Services:** Reusable blocks of code (like data services or utility functions) that are independent of components.
- **Directives:** Extend HTML with new syntax, attributes, or elements to encapsulate and reuse UI behavior.
- **Data Binding:** Synchronizes data between the component's class and the template.
- **Dependency Injection:** Provides a framework for supplying classes and values to other classes.
- **Routing:** Defines how the application navigates between different views.

53. What are Directives?

- **Directives** are classes that add new behavior to existing DOM elements.
- They can modify an element's appearance or behavior, create custom elements, or respond to events.
- Directives are of Three types: Attribute Directives, Structural Directives, and Custom Directives.
- Examples: *ngIf, *ngFor, [class.active].

54. What are Components?

- **Components** are the fundamental building blocks of an Angular application's UI.
- Each component has its own template, which defines the HTML structure of the component's view.
- Components encapsulate both the UI and the logic that drives it.

55. What are the differences between Component and Directive?

Feature	Component	Directive
Purpose	Represents a self-contained UI element with its own template and logic.	Modifies the behavior of existing DOM elements.
Structure	Has a template and a class.	Usually just a class.
Example	<app-product-list>	*ngIf, [class.active]

56. What is a template?

A **template** is the HTML that defines the view of a component.

- It can contain:
    - Regular HTML elements
    - Angular template syntax (e.g., data binding, directives)
    - Placeholders for component inputs and outputs

57. What is a Module?

- A **module** is a container for a cohesive block of functionality, such as a feature area or a group of related services.

- It defines a boundary for the application, controlling what parts of the application should be visible to other parts.
- The root module of an Angular application is typically called AppModule.

10) How does an Angular application work?
11) What are lifecycle hooks available?
12) What is a data binding?
13) What is metadata?
14) What is Angular CLI?
15) What is the difference between constructor and ngOnInit?
16) What is a service
17) What is dependency injection in Angular?
18) How is Dependency Hierarchy formed?
19) What is the purpose of async pipe?
20) What is the option to choose between inline and external template file?
21) What is the purpose of *ngFor directive?
22) What is the purpose of ngIf directive?
23) What happens if you use script tag inside template?
24) What is interpolation?
25) What are template expressions?
26) What are template statements?
27) How do you categorize data binding types?
28) What are pipes?
29) What is a parameterized pipe?
30) How do you chain pipes?
31) What is a custom pipe?
32) Give an example of custom pipe?
33) What is the difference between pure and impure pipe?
34) What is a bootstrapping module?
35) What are observables?
36) What is HttpClient and its benefits?
37) Explain on how to use HttpClient with an example?
38) How can you read full response?
39) How do you perform Error handling?
40) What is RxJS?
41) What is subscribing?
42) What is an observable?
43) What is an observer?
44) What is the difference between promise and observable?
45) What is multicasting?

46) How do you perform error handling in observables?
47) What is the shorthand notation for subscribe method?
48) What are the utility functions provided by RxJS?
49) What are observable creation functions?
50) What will happen if you do not supply handler for the observer?
51) What are Angular elements?
52) What is the browser support of Angular Elements?
53) What are custom elements?
54) Do I need to bootstrap custom elements?
55) Explain how custom elements works internally?
56) How to transfer components to custom elements?
57) What are the mapping rules between Angular component and custom element?
58) How do you define typings for custom elements?
59) What are dynamic components?
60) What are the various kinds of directives?
61) How do you create directives using CLI?
62) Give an example for attribute directives?
63) What is Angular Router?
64) What is the purpose of base href tag?
65) What are the router imports?
66) What is router outlet?
67) What are router links?
68) What are active router links?
69) What is router state?
70) What are router events?
71) What is activated route?
72) How do you define routes?
73) What is the purpose of Wildcard route?
74) Do I need a Routing Module always?
75) What is Angular Universal?
76) What are different types of compilation in Angular?
77) What is JIT?
78) What is AOT?
79) Why do we need compilation process?
80) What are the advantages with AOT?
81) What are the ways to control AOT compilation?
82) What are the restrictions of metadata?
83) What are the three phases of AOT?
84) Can I use arrow functions in AOT?
85) What is the purpose of metadata json files?

86) Can I use any javascript feature for expression syntax in AOT?
87) What is folding?
88) What are macros?
89) Give an example of few metadata errors?
90) What is metadata rewriting?
91) How do you provide configuration inheritance?
92) How do you specify angular template compiler options?
93) How do you enable binding expression validation?
94) What is the purpose of any type cast function?
95) What is Non null type assertion operator?
96) What is type narrowing?
97) How do you describe various dependencies in angular application?
98) What is zone?
99) What is the purpose of common module?
100) What is codelyzer?
101) What is angular animation?
102) What are the steps to use animation module?
103) What is State function?
104) What is Style function?
105) What is the purpose of animate function?
106) What is transition function?
107) How to inject the dynamic script in angular?
108) What is a service worker and its role in Angular?
109) What are the design goals of service workers?
110) What are the differences between AngularJS and Angular with respect to dependency injection?
111) What is Angular Ivy?
112) What are the features included in ivy preview?
113) Can I use AOT compilation with Ivy?
114) What is Angular Language Service?
115) How do you install angular language service in the project?
116) Is there any editor support for Angular Language Service?
117) Explain the features provided by Angular Language Service?
118) How do you add web workers in your application?
119) What are the limitations with web workers?
120) What is Angular CLI Builder?
121) What is a builder?
122) How do you invoke a builder?
123) How do you create app shell in Angular?
124) What are the case types in Angular?

125) What are the class decorators in Angular?
126) What are class field decorators?
127) What is declarable in Angular?
128) What are the restrictions on declarable classes?
129) What is a DI token?
130) What is Angular DSL?
131) What is an rxjs Subject?
132) What is Bazel tool?
133) What are the advantages of Bazel tool?
134) How do you use Bazel with Angular CLI?
135) How do you run Bazel directly?
136) What is platform in Angular?
137) What happens if I import the same module twice?
138) How do you select an element with in a component template?
139) How do you detect route change in Angular?
140) How do you pass headers for HTTP client?
141) What is the purpose of differential loading in CLI?
142) Is Angular supports dynamic imports?
143) What is lazy loading?
144) What are workspace APIs?
145) How do you upgrade angular version?
146) What is Angular Material?
147) How do you upgrade location service of angularjs?
148) What is NgUpgrade?
149) How do you test Angular application using CLI?
150) How to use polyfills in Angular application?
151) What are the ways to trigger change detection in Angular?
152) What are the differences of various versions of Angular?
153) What are the security principles in angular?
154) What is the reason to deprecate Web Tracing Framework?
155) What is the reason to deprecate web worker packages?
156) How do you find angular CLI version?
157) What is the browser support for Angular?
158) What is schematic
159) What is rule in Schematics?
160) What is Schematics CLI?
161) What are the best practices for security in angular?
162) What is Angular security model for preventing XSS attacks?
163) What is the role of template compiler for prevention of XSS attacks?
164) What are the various security contexts in Angular?

165) What is Sanitization? Is angular supports it?
166) What is the purpose of innerHTML?
167) What is the difference between interpolated content and innerHTML?
168) How do you prevent automatic sanitization?
169) Is safe to use direct DOM API methods in terms of security?
170) What is DOM sanitizer?
171) How do you support server side XSS protection in Angular application?
172) Is angular prevents http level vulnerabilities?
173) What are Http Interceptors?
174) What are the applications of HTTP interceptors?
175) Is multiple interceptors supported in Angular?
176) How can I use interceptor for an entire application?
177) How does Angular simplifies Internationalization?
178) How do you manually register locale data?
179) What are the four phases of template translation?
180) What is the purpose of i18n attribute?
181) What is the purpose of custom id?
182) What happens if the custom id is not unique?
183) Can I translate text without creating an element?
184) How can I translate attribute?
185) List down the pluralization categories?
186) What is select ICU expression?
187) How do you report missing translations?
188) How do you provide build configuration for multiple locales?
189) What is an angular library?
190) What is AOT compiler?
191) How do you select an element in component template?
192) What is TestBed?
193) What is protractor?
194) What is collection?
195) How do you create schematics for libraries?
196) How do you use jquery in Angular?
197) What is the reason for No provider for HTTP exception?
198) What is router state?
199) How can I use SASS in angular project?
200) What is the purpose of hidden property?
201) What is the difference between ngIf and hidden property?
202) What is slice pipe?
203) What is index property in ngFor directive?
204) What is the purpose of ngFor trackBy?

205) What is the purpose of ngSwitch directive?
206) Is it possible to do aliasing for inputs and outputs?
207) What is safe navigation operator?
208) Is any special configuration required for Angular9?
209) What are type safe TestBed API changes in Angular9?
210) Is mandatory to pass static flag for ViewChild?
211) What are the list of template expression operators?
212) What is the precedence between pipe and ternary operators?
213) What is an entry component?
214) What is a bootstrapped component?
215) How do you manually bootstrap an application?
216) Is it necessary for bootstrapped component to be entry component?
217) What is a routed entry component?
218) Why is not necessary to use entryComponents array every time?
219) Do I still need to use entryComponents array in Angular9?
220) Is it all components generated in production build?
221) What is Angular compiler?
222) What is the role of ngModule metadata in compilation process?
223) How does angular finds components, directives and pipes?
224) Give few examples for NgModules?
225) What are feature modules?
226) What are the imported modules in CLI generated feature modules?
227) What are the differences between ngmodule and javascript module?
228) What are the possible errors with declarations?
229) What are the steps to use declaration elements?
230) What happens if browserModule used in feature module?
231) What are the types of feature modules?
232) What is a provider?
233) What is the recommendation for provider scope?
234) How do you restrict provider scope to a module?
235) How do you provide a singleton service?
236) What are the different ways to remove duplicate service registration?
237) How does forRoot method helpful to avoid duplicate router instances?
238) What is a shared module?
239) Can I share services using modules?
240) How do you get current direction for locales??
241) What is ngcc?
242) What classes should not be added to declarations?
243) What is ngzone?
244) What is NoopZone?

245) How do you create displayBlock components?
246) What are the possible data change scenarios for change detection?
247) What is a zone context?
248) What are the lifecycle hooks of a zone?
249) Which are the methods of NgZone used to control change detection?
250) How do you change the settings of zonejs?
251) How do you trigger an animation?
252) How do you configure injectors with providers at different levels?
253) Is it mandatory to use injectable on every service class?
254) What is an optional dependency?
255) What are the types of injector hierarchies?
256) What are reactive forms?
257) What are dynamic forms?
258) What are template driven forms?
259) What are the differences between reactive forms and template driven forms?
260) What are the different ways to group form controls?
261) How do you update specific properties of a form model?
262) What is the purpose of FormBuilder?
263) How do you verify the model changes in forms?
264) What are the state CSS classes provided by ngModel?
265) How do you reset the form?
266) What are the types of validator functions?
267) Can you give an example of built-in validators?
268) How do you optimize the performance of async validators?
269) How to set ngFor and ngIf on the same element?
270) What is host property in css?
271) How do you get the current route?
272) What is Component Test Harnesses?
273) What is the benefit of Automatic Inlining of Fonts?
274) What is content projection?
275) What is ng-content and its purpose?
276) What is standalone component?
277) How to create a standalone component uing CLI command?
278) How to create a standalone component manually?
279) What is hydration ?
280) What are Angular Signals?
281) Explain Angular Signals with an example
282) What are the Route Parameters? Could you explain each of them?