Certainly! Here are brief answers to the selected questions:

1. What is a Module, and what does it contain?

- A Module in Angular is a container that groups related components, directives, services, and other code files. It helps organize and modularize an Angular application. It contains declarations, providers, imports, exports, and other configuration metadata.

2. What is Routing Guard in Angular?

- Routing Guards are used in Angular to control and protect navigation to certain routes. They provide functionality to allow or prevent a user from navigating to a specific route based on certain conditions. There are different types of routing guards, including CanActivate, CanDeactivate, CanLoad, and Resolve.

3. What is the equivalent of ngShow and ngHide in Angular?

- In Angular, the equivalent of ngShow is *ngIf, which conditionally renders or hides elements based on a given expression. For ngHide, the equivalent is [hidden], which applies a CSS style to hide elements based on a condition.

4. What is the difference between @Component and @Directive in Angular?

- @Component is a decorator used to define a component in Angular. It represents a reusable UI element with its own template, styles, and behavior.

- @Directive is a decorator used to define a directive in Angular. It allows you to extend the behavior of existing elements or create new reusable behaviors without templates.

5. What is a Component? Why would you use it?

- A Component is a fundamental building block in Angular that represents a part of the user interface with its own template, styles, and logic. Components encapsulate related functionality and provide reusability, maintainability, and separation of concerns in an application.

6. What are Observables?

- Observables are a key concept in reactive programming and are extensively used in Angular. They represent a stream of data over time that can be observed by subscribers. Observables are used for asynchronous operations, event handling, and handling data streams in Angular applications.

7. What is a Service, and when will you use it?

- A Service in Angular is a class that provides a specific functionality, such as data manipulation, API calls, or sharing data between components. Services are commonly used to keep the application logic separate from the components and enable code reuse and maintainability.

8. What's the difference between an Angular Component and Module?

- An Angular Component represents a reusable UI element with its own template, styles, and behavior. It encapsulates the presentation logic and interacts with the data.

- An Angular Module is a container that groups related components, directives, services, and other code files. It helps organize and modularize an Angular application by providing a context for the components.

9. How do you categorize data binding types?

- In Angular, data binding types can be categorized as follows:

  - Interpolation: Used to bind a value directly into the template using double curly braces, e.g., {{ variable }}.

  - Property binding: Used to set the value of an element's property using square brackets, e.g., [property]="value".

  - Event binding: Used to listen to events and trigger a function in response using parentheses, e.g., (event)="function()".

  - Two-way binding: Combines property and event binding, allowing data to flow both from the component to the template and vice versa using the ngModel directive.

10. When would you use eager module loading?

- Eager module loading is used when a module needs to be loaded synchronously when the application starts. It is suitable for smaller modules or when the module is required for the initial rendering of the application. Eager loading ensures that the module is available immediately when needed.

11. Explain how Custom Elements work internally.

- Custom Elements are a part of the Web Components standard and allow developers to create their own reusable HTML elements with custom behavior. Internally, Custom Elements are created using the `customElements` API in the browser. Developers define a custom element by extending the `HTMLElement` class and providing the desired behavior and properties. Once registered, the custom element can be used in HTML like any other built-in element, and the browser handles its instantiation and rendering.

12. What is Multicasting?

- Multicasting refers to the ability of an Observable in RxJS to have multiple subscribers receiving the same sequence of values emitted by the Observable. It allows for the efficient sharing of values among multiple subscribers, reducing unnecessary duplicate computations or API calls. Multicasting is commonly achieved using operators like `share`, `publish`, or `multicast` in RxJS.

13. What is the difference between declarations, providers, and imports in NgModule?

- `Declarations`: The `declarations` property in `NgModule` is used to specify the components, directives, and pipes that belong to the module. It allows Angular to recognize and use these items within the module. The declarations array should include the classes of components, directives, and pipes that are part of the current module.

- `Providers`: The `providers` property in `NgModule` is used to register services or other dependencies at the module level. It allows the provided services to be shared and accessible across the module and its components. Services registered at the module level create a single instance shared by all components within the module and its child modules.

- `Imports`: The `imports` property in `NgModule` is used to import other modules that are required by the current module. It allows the exported components, directives, and pipes from the imported module to be used within the current module. The imported module's exports become part of the current module's scope, enabling the usage of those exported items.

For example, if you have a custom component, `MyComponent,` that needs to be used within the current module, you would include it in the `declarations` array. If you have a service `MyService` that needs to be shared across the module, you would include it in the `providers` array. Finally, if you need to use features from another module, such as `FormsModule`, you would import it using the `imports` array.

14. What's new in Angular 6 and why should we upgrade to it?

- Some notable features in Angular 6 include the introduction of the Angular Elements package, improved Angular CLI commands, support for the Angular Material library, and updates to the Angular Material Design CDK. Additionally, Angular 6 provides enhanced tree shaking capabilities, performance improvements, and bug fixes. Upgrading to Angular 6 offers access to these new features, improved tooling, and potential performance optimizations.

15. What is AOT (Ahead of Time) compilation?

- AOT (Ahead of Time) compilation is a process in Angular where the Angular templates are compiled during the build step rather than at runtime. This compilation process transforms the templates into highly optimized JavaScript code, resulting in faster application startup, smaller

bundle sizes, and improved runtime performance. AOT compilation eliminates the need for the Angular compiler in the production bundle, making the application more efficient.

16. Explain the difference between Constructor and ngOnInit.

- The constructor is a method defined in a class that is called when an instance of the class is created. It is used for initializing the object and setting up dependencies. The constructor is called before the ngOnInit lifecycle hook.

- ngOnInit is a lifecycle hook in Angular that is called after the constructor and is specifically used for initialization tasks. It is a good place to fetch data from services, initialize component properties, or perform any setup logic. ngOnInit is called once, immediately after the first ngOnChanges.

17. What is a Parameterized pipe?

- A Parameterized pipe is a custom pipe in Angular that accepts additional arguments, known as parameters. These parameters are passed to the pipe in the template using a colon (:) followed by the parameter value. Parameterized pipes allow for more flexibility and customization, as the behavior of the pipe can be modified based on the provided parameters.

18. Explain Lazy Loading in Angular.

- Lazy Loading is a feature in Angular that allows the loading of modules and their associated components only when they are needed, rather than loading them all upfront during the initial app load. Lazy loading helps optimize application performance by reducing the initial load time and only fetching the necessary code when a specific route is accessed. It improves the user experience by delivering faster navigation and reducing the overall bundle size of the application.

19. How would you insert an embedded view from a prepared TemplateRef?

- To insert an embedded view from a prepared TemplateRef, you would use the `createEmbeddedView` method provided by the `ViewContainerRef` class. First, you obtain a reference to the view container where you want to insert the embedded view. Then, you create the embedded view using the `createEmbeddedView` method, passing the prepared TemplateRef as an argument. Finally, you can manipulate the embedded view, bind data to it, and insert it into the view container using methods like `attach` or `insert`.

20. What are the lifecycle hooks for components and directives?

- The lifecycle hooks available for components and directives in Angular are as follows:

  - `ngOnChanges`: Called when one or more input properties change.

  - `ngOnInit`: Called once after the first `ngOnChanges` when the component or directive is initialized.

- `ngDoCheck`: Called during every change detection run.

- `ngAfterContentInit`: Called once after the component or directive content is initialized.

- `ngAfterContentChecked`: Called after every change detection run when the component or directive content has been checked.

- `ngAfterViewInit`: Called once after the component or directive's view has been initialized.

- `ngAfterViewChecked`: Called after every change detection run when the component or directive's view has been checked.

- `ngOnDestroy`: Called once when the component or directive is about to be destroyed.

21. What is the difference between pure and impure pipe?

- In Angular, pure and impure are two types of pipes:

- A pure pipe is a pipe that is stateless and doesn't change its output based on mutable data. It means that a pure pipe's output is only recalculated if its input value changes. Pure pipes are optimized for performance and should not have any side effects.

- An impure pipe is a pipe that can have side effects and recalculate its output on every change detection cycle, regardless of whether the input value has changed. Impure pipes are not as performant as pure pipes and should be used with caution.

22. How to detect a route change in Angular?

- To detect a route change in Angular, you can subscribe to the `Router` service's events. The `Router` provides an `events` observable that emits various events, including `NavigationStart`, `NavigationEnd`, `NavigationCancel`, and `NavigationError`. You can subscribe to the `events` observable and perform actions based on the specific event type, such as updating component data or performing side effects when a route changes.

23. Name some security best practices in Angular.

- Some security best practices in Angular include:

- Implementing proper input validation and sanitization to prevent cross-site scripting (XSS) attacks.

- Applying appropriate server-side validation and authorization checks to protect sensitive data and prevent unauthorized access.

- Guarding against code injection attacks by using safe navigation operators (`?.`) and sanitizing user-generated content.

- Using secure HTTP protocols (HTTPS) to protect data transmission between the client and server.

- Employing appropriate authentication and authorization mechanisms, such as token-based authentication or OAuth.

- Regularly updating Angular and its dependencies to benefit from security patches and fixes.

- Following secure coding practices, such as avoiding inline scripts, using strong passwords, and adhering to the principle of least privilege.

24. What are the advantages of using AOT compilation?

- Using AOT (Ahead of Time) compilation in Angular provides several advantages, including:

- Faster application startup time: AOT compiles templates during the build process, resulting in faster rendering and reduced loading times.

- Smaller bundle sizes: AOT eliminates the need for the Angular compiler in the production bundle, reducing the overall bundle size and improving network performance.

- Template syntax validation: AOT compilation performs template syntax checks during the build, catching potential errors before runtime.

- Better runtime performance: AOT-compiled code is optimized for production, resulting in improved runtime performance compared to Just-in-Time (JIT) compilation.

- Enhanced security: AOT compilation removes the need to ship the Angular compiler to the client, reducing the risk of code tampering or injection attacks.

25. What does detectChanges do in Angular Jasmine tests?

- In Angular Jasmine tests, the `detectChanges` method is used to trigger change detection manually. Change detection is the process that Angular uses to update the UI when there are changes to the component's data or state. The `detectChanges` method forces the execution of change detection for the component and its child components, ensuring that any changes are reflected in the view. It is often called after making changes to component properties or triggering events in tests to verify the expected behavior.

26. Why would you use lazy loading modules in an Angular app?

- Lazy loading modules in an Angular app offer several benefits, including:

- Improved performance: Lazy loading allows you to load modules on demand when they are needed, reducing the initial bundle size and improving the application's loading speed. It avoids loading unnecessary modules upfront, resulting in a faster initial page load for the user.

- Faster application startup: With lazy loading, only the required modules are loaded initially, enabling the application to start up more quickly. This is particularly useful for larger applications with numerous modules, as it avoids the overhead of loading the entire application at once.

- Optimal resource utilization: Lazy loading helps in resource management by loading modules asynchronously as the user navigates through the application. It allows the application to load modules in the background while the user interacts with the already loaded parts, optimizing resource utilization.

- Better user experience: By reducing the initial load time and improving the overall performance, lazy loading enhances the user experience. Users can start interacting with the application sooner and navigate between different sections seamlessly.

- Code modularization: Lazy loading promotes modularization of the codebase. Each module can be developed and maintained independently, making it easier to manage large applications and enabling teams to work on separate modules without conflicts.

26. Do I need to bootstrap custom elements?

- No, you do not need to bootstrap custom elements explicitly in Angular. Angular automatically handles the bootstrapping process for Angular components. However, if you want to use a custom element within an Angular application, you need to define a directive that wraps the custom element and include it in your Angular module's declarations.

27. Why would you use lazy loading modules in an Angular app?

- Lazy loading modules in an Angular app are useful for improving performance by loading modules and their associated components only when they are needed. This reduces the initial load time of the application, as it doesn't have to load all modules upfront. Lazy loading allows for more efficient use of network resources and enables faster navigation within the application.

28. Why would you use renderer methods instead of using native element methods?

- Using renderer methods instead of native element methods in Angular provides better abstraction and ensures compatibility across different platforms. The renderer methods provide a layer of abstraction that allows Angular to work seamlessly with various rendering targets, such as the browser, server-side rendering, or native mobile platforms. By using renderer methods, you ensure that your code remains platform-independent and can be executed consistently across different environments.

29. What is ngUpgrade?

- ngUpgrade is a library provided by Angular that allows you to gradually upgrade an AngularJS (Angular 1.x) application to Angular (Angular 2.x and beyond). It provides a bridge that enables both AngularJS and Angular code to coexist within the same application. With ngUpgrade, you can incrementally migrate components, services, and other code from AngularJS to Angular, making the migration process smoother and more manageable.

30. What is Angular Router?

- Angular Router is a module in Angular that provides a powerful routing system for building single-page applications. It enables navigation between different views and components based on URL changes. Angular Router allows you to define routes, configure route parameters, guard routes,

handle redirects, and perform lazy loading of modules. It plays a vital role in creating a navigational structure and providing a seamless user experience within an Angular application.

31. How would you control the size of an element on the resize of the window in a component?

- To control the size of an element on window resize in an Angular component, you can utilize the `HostListener` decorator along with the `window:resize` event. You can define a method in your component annotated with `@HostListener('window:resize')` to listen to the window resize event. Within this method, you can update the size or dimensions of the element by accessing it through the component's ViewChild or ElementRef.

32. How do you perform error handling in Observables?

- In Observables, you can perform error handling using the `catchError` operator provided by RxJS. The `catchError` operator allows you to intercept errors emitted by an Observable and handle them gracefully. You can provide a callback function to `catchError` that handles the error and returns a new Observable or throws a custom error. This enables you to handle errors, perform fallback actions, or transform the error stream as needed within the Observable pipeline.

33. What is Angular Universal?

- Angular Universal is a server-side rendering (SSR) solution provided by Angular. It allows you to generate static HTML pages on the server and send them to the client, enhancing the initial load time and improving search engine optimization (SEO). Angular Universal runs the Angular application on the server and renders the initial view, which is then hydrated on the client side. This enables faster perceived performance and better accessibility for users and search engines.

34. What is the difference between declarations, providers, and import in NgModule?

- In an NgModule decorator:

  - `declarations` is used to specify the components, directives, and pipes that belong to the module. They are declared so that they can be used within the module.

  - `providers` is used to register services or other dependencies at the module level. It allows the provided services to be shared and accessible across the module and its components.

  - `imports` is used to import other modules that are required by the current module. It makes the exported components, directives, and pipes of the imported module available within the current module.

35. What's new in Angular 6 and why should we upgrade to it?

- Some notable features in Angular 6 include the introduction of the Angular Elements package, improved Angular CLI commands, support for the Angular Material library, and updates to the Angular Material Design CDK. Additionally, Angular 6 provides enhanced tree shaking capabilities,

performance improvements, and bug fixes. Upgrading to Angular 6 offers access to these new features, improved tooling, and potential performance optimizations.

36. What is AOT (Ahead of Time) compilation?

- AOT (Ahead of Time) compilation is a process in Angular where the Angular templates are compiled during the build step rather than at runtime. This compilation process transforms the templates into highly optimized JavaScript code, resulting in faster application startup, smaller bundle sizes, and improved runtime performance. AOT compilation eliminates the need for the Angular compiler in the production bundle, making the application more efficient.

37. Explain the difference between Constructor and ngOnInit.

- The constructor is a method defined in a class that is called when an instance of the class is created. It is used for initializing the object and setting up dependencies. The constructor is called before the ngOnInit lifecycle hook.

- ngOnInit is a lifecycle hook in Angular that is called after the constructor and is specifically used for initialization tasks. It is a good place to fetch data from services, initialize component properties, or perform any setup logic. ngOnInit is called once, immediately after the first ngOnChanges.

38. What is a Parameterized pipe?

- A Parameterized pipe is a custom pipe in Angular that accepts additional arguments, known as parameters. These parameters are passed to the pipe in the template using a colon (:) followed by the parameter value. Parameterized pipes allow for more flexibility and customization, as the behavior of the pipe can be modified based on the provided parameters.

39. Explain Lazy Loading in Angular.

- Lazy Loading is a feature in Angular that allows the loading of modules and their associated components only when they are needed, rather than loading them all upfront during the initial app load. Lazy loading helps optimize application performance by reducing the initial load time and only fetching the necessary code when a specific route is accessed. It improves the user experience by delivering faster navigation and reducing the overall bundle size of the application.

40. How would you insert an embedded view from a prepared TemplateRef?

- To insert an embedded view from a prepared TemplateRef, you would use the `createEmbeddedView` method provided by the `ViewContainerRef` class. First, you obtain a reference to the view container where you want to insert the embedded view. Then, you create the embedded view using the `createEmbeddedView` method, passing the prepared TemplateRef as an argument. Finally, you can manipulate the embedded view, bind data to it, and insert it into the view container using methods like `attach` or `insert`.

41. What are the lifecycle hooks for components and directives?

- The lifecycle hooks available for components and directives in Angular are as follows:

  - `ngOnChanges`: Called when one or more input properties change.

  - `ngOnInit`: Called once after the first `ngOnChanges` when the component or directive is initialized.

  - `ngDoCheck`: Called during every change detection run.

  - `ngAfterContentInit`: Called once after the component or directive content is initialized.

  - `ngAfterContentChecked`: Called after every change detection run when the component or directive content has been checked.

  - `ngAfterViewInit`: Called once after the component or directive's view has been initialized.

  - `ngAfterViewChecked`: Called after every change detection run when the component or directive's view has been checked.

  - `ngOnDestroy`: Called once when the component or directive is about to be destroyed.

42. What is the difference between pure and impure pipe?

- In Angular, pure and impure are two types of pipes:

  - A pure pipe is a pipe that is stateless and doesn't change its output based on mutable data. It means that a pure pipe's output is only recalculated if its input value changes. Pure pipes are optimized for performance and should not have any side effects.

  - An impure pipe is a pipe that can have side effects and recalculate its output on every change detection cycle, regardless of whether the input value has changed. Impure pipes are not as performant as pure pipes and should be used with caution.

43. How to detect a route change in Angular?

- To detect a route change in Angular, you can subscribe to the `Router` service's events. The `Router` provides an `events` observable that emits various events, including `NavigationStart`, `NavigationEnd`, `NavigationCancel`, and `NavigationError`. You can subscribe to the `events` observable and perform actions based on the specific event type, such as updating component data or performing side effects when a route changes.

44. Name some security best practices in Angular.

- Some security best practices in Angular include:

  - Implementing proper input validation and sanitization to prevent cross-site scripting (XSS) attacks.

- Applying appropriate server-side validation and authorization checks to protect sensitive data and prevent unauthorized access.

- Guarding against code injection attacks by using safe navigation operators (`?.`) and sanitizing user-generated content.

- Using secure HTTP protocols (HTTPS) to protect data transmission between the client and server.

- Employing appropriate authentication and authorization mechanisms, such as token-based authentication or OAuth.

- Regularly updating Angular and its dependencies to benefit from security patches and fixes.

- Following secure coding practices, such as avoiding inline scripts, using strong passwords, and adhering to the principle of least privilege.

45. What are the advantages of using AOT (Ahead of Time) compilation?

- Using AOT (Ahead of Time) compilation in Angular provides several advantages, including:

- Faster application startup time: AOT compiles templates during the build process, resulting in faster rendering and reduced loading times.

- Smaller bundle sizes: AOT eliminates the need for the Angular compiler in the production bundle, reducing the overall bundle size and improving network performance.

- Template syntax validation: AOT compilation performs template syntax checks during the build, catching potential errors before runtime.

- Better runtime performance: AOT-compiled code is optimized for production, resulting in improved runtime performance compared to Just-in-Time (JIT) compilation.

- Enhanced security: AOT compilation removes the need to ship the Angular compiler to the client, reducing the risk of code tampering or injection attacks.

- Improved SEO (Search Engine Optimization): AOT generates static HTML files that can be pre-rendered on the server, making the app more SEO-friendly and improving search engine discoverability.

- Better offline and mobile support: AOT allows for the creation of Progressive Web Apps (PWAs) that can work offline and offer improved performance on mobile devices.

- Compatibility with strict mode and tree shaking: AOT enables compatibility with strict mode, enabling stricter type checking and better optimization through tree shaking, resulting in smaller and more efficient bundles.