ANGULAR INTERVIEW QUESTIONS

1

What is RxJs ?

RxJS is stand for Reactive Extension for JavaScript. It is a JavaScript library used to work with Reactive programming that works with asynchronous data calls, callbacks, and event-based programs. RxJS can be used with other JavaScript libraries like Angular, ReactJS, Vue.js, Node.js, etc. Both JavaScript and TypeScript support it.

Some features of RxJs:

- **Observable**: An observable is a function that creates an observer and attaches it to the source where values are expected, for example, clicks, mouse events from a Dom element or an Http request, etc.
- **Observer**: It is an object with next(), error() and complete() methods, that will get called when there is interaction to the with the observable i.e., the source interacts for an example button click, Http request, etc.
- **Subscription**: When the observable is created, to execute the observable we need to subscribe to it. It can also be used to cancel the execution
- **Operators**: An operator is a pure function that takes in observable as input and the output is also an observable.
- **Subject**: A subject is an observable that can multicast i.e., talk to many observers.

2

What is Subject ?

An RxJS Subject is a special type of Observable that allows values to be multicasted to many Observers. While plain Observables are unicast (each subscribed Observer owns an independent execution of the Observable), Subjects are multicast.

There are mainly four variants of RxJS subjects:

- **Subject**: This is the standard RxJS Subject. It doesn't have any initial value or replay behaviour.
- **BehaviorSubject** : This variant of RxJS subject requires an initial value and emits its current value (last emitted item) to new subscribers.
- **ReplaySubject** : This variant of RxJS subject is used to emit a specified number of last emitted values (a replay) to new subscribers.
- **AsyncSubject** : The AsyncSubject emits the latest value to observers upon completion.

3

What is Angular element ?

Angular Elements is a feature of the Angular framework that allows you to package Angular components as standalone, reusable web components. These web components can be used in different web applications, even those that are not built with Angular. Angular Elements makes it easier to share and integrate Angular components in various projects, providing a way to encapsulate and distribute Angular functionality as self-contained, custom HTML elements.

To implement Angular Elements in Angular, follow these steps:

- **Create an Angular component**: Develop the Angular component you want to export as an Angular Element.
- **Configure Angular Elements**: In your Angular project, configure Angular Elements by importing the necessary packages and modifying your Angular application to be element-ready.
- **Define your custom element**: Use Angular's **createCustomElement** function to create a custom web component from your Angular component.
- **Define hub event handlers**: Define event handlers in your service to respond to events emitted from the SignalR hub. These handlers can update your Angular components with real-time data.
- **Register the custom element**: Register your custom element using the **customElements** API, making it available as an HTML tag.
- **Build your Angular Element**: Build your Angular project to produce a JavaScript file that contains your Angular Element.
- **Include the Angular Element in other projects**: You can now use the generated JavaScript file in other web applications, regardless of the framework they are built with, by including it as a script tag. Your Angular Element can be used as a custom HTML element in these projects.

4

What is Dynamic component ?

In Angular, a dynamic component is a component that is created and rendered dynamically at runtime. This means that you can programmatically generate, load, and display components in your application based on specific conditions, user interactions, or data. Dynamic components are useful for scenarios where you need to add or remove parts of your UI on the fly, such as creating dynamic forms, dialog boxes, or widgets. They offer flexibility and adaptability in your Angular applications.

To implement Angular Elements in Angular, follow these steps:

- **Create the dynamic component**: Define the component you want to create dynamically, typically with its own template, styles, and functionality.
- **Use ComponentFactoryResolver**: Inject the **ComponentFactoryResolver** service into your Angular component or service. This service allows you to obtain a **ComponentFactory** for the dynamic component you want to create.
- **Create a ViewContainerRef**: In your template, add a **ViewContainerRef** where you want to insert the dynamic component. This serves as a placeholder for the new component.
- **Create and insert the component**: Using the **ComponentFactory**, you can create an instance of your dynamic component and insert it into the **ViewContainerRef**. You can also pass data to the dynamic component if needed.
- **Destroy or remove the component**: If required, you can also remove the dynamic component from the **ViewContainerRef** when it's no longer needed.

5

What is SignalR ?

SignalR is a real-time web communication library that allows bi-directional communication between a server and clients in web applications. In the context of Angular, SignalR can be used to enable real-time features like chat, notifications, and live updates in your web applications by facilitating seamless communication between the Angular frontend and a SignalR server. It's a powerful tool for building interactive and responsive web applications.

To implement SignalR in an Angular application, you can follow these steps:

- **Install SignalR client library for JavaScript**: Use npm or yarn to install the SignalR client library for JavaScript in your Angular project.
- **Create a SignalR service**: Create a service in your Angular app to manage SignalR connections. This service will handle the initialization, connection, and event handling for SignalR.Avoid using ngIf with complex expressions
- **Initialize SignalR connection**: In your service, establish a connection to the SignalR hub on the server by providing the hub URL.
- **Define hub event handlers**: Define event handlers in your service to respond to events emitted from the SignalR hub. These handlers can update your Angular components with real-time data.
- **Connect and disconnect**: Implement methods in your service to connect and disconnect from the SignalR hub as needed.
- **Inject the service into Angular components**: Inject the SignalR service into your Angular components that need real-time communication and subscribe to the events or data provided by the service.
- **Use real-time features**: With the SignalR service in place, you can now use real-time features in your Angular components, such as live updates, chat, or notifications.

6

What is standalone components ?

In Angular, standalone components are components that are not directly embedded within the main application. Instead, they can function independently and be reused in various parts of the application or even in different Angular applications. Standalone components are typically designed to have their own self-contained functionality and can be shared and used as building blocks in different contexts, enhancing code modularity and reusability.

Here are the top 5 features of standalone components:

- **Reusability:** Standalone components can be reused in different parts of the same application or even across multiple Angular applications. This promotes code reusability and reduces duplication.
- **Encapsulation:** Standalone components encapsulate their functionality, styles, and templates. This encapsulation makes it easier to maintain and understand the component's behavior and appearance.
- **Isolation:** Standalone components are independent, meaning they don't depend on the specific context of the application where they are used. This isolation simplifies testing and reduces potential side effects on other parts of the application.
- **Modularity:** Standalone components contribute to the modularity of your application. They can be individually developed, tested, and updated without affecting other parts of the application.
- **Consistency:** Using standalone components promotes consistency in the user interface and functionality. You can maintain a consistent look and behavior by reusing the same component in multiple places.

7

What is view encapsulation ?

Encapsulation is one of the core ideas of object-oriented programming (OOP). It establishes the principle of maintaining all data and methods that interact with that data in a single unit (or class). It is like hiding details of the implementation details from the public. The consumer of encapsulated object know that it works but does not know how it works.

The following three strategies are provided by Angular to determine how styles are applied:

- ViewEncapsulation.None
- ViewEncapsulation.Emulated
- ViewEncapsulation.ShadowDom

8

What is Change detection ?

Change detection is a mechanism used in frameworks like Angular to automatically detect and update the user interface (UI) when the underlying data or model changes. It ensures that any modifications to the application's state are reflected in the view, allowing for real-time updates and synchronization between the data and what the user sees on the screen. This process involves checking for changes in the data and propagating those changes to the UI elements, making the application responsive and interactive.

Here are the top 5 features of Angular's change detection mechanism:

- **Automatic Data Binding:** Angular's change detection automatically binds data from the component to the view. When the component's data changes, the view is automatically updated to reflect those changes. This simplifies the process of keeping the UI in sync with the application's state.
- **Zones:** Angular uses zones to track asynchronous operations and automatically trigger change detection when asynchronous tasks complete. This means that you don't need to manually trigger change detection after asynchronous operations like HTTP requests.
- **Change Detection Strategy:** Angular allows you to specify the change detection strategy for a component. You can choose between two strategies: default (checking the entire component tree) or OnPush (checking only the components where the data has changed), optimizing performance in specific scenarios.
- **Change Detection Tree:** Angular maintains a tree structure of components, which allows it to efficiently track and propagate changes in the component hierarchy. Only components that are affected by a change are updated, reducing unnecessary work.
- **Optimization and Performance:** Angular's change detection is optimized for performance. It minimizes the number of DOM updates by batching changes and only updating the view when needed. This results in fast and efficient UI updates.

9

What is Ngrx ?

NgRx is a state management library for Angular applications. It is based on the Redux pattern and provides a predictable and centralized way to manage the state of your Angular application. NgRx helps you maintain a single source of truth for your application's data, manage complex application state, and handle data flow in a more structured and maintainable manner. It is often used for handling state in large and complex Angular applications, offering features like actions, reducers, and effects to manage and update the application's data and user interface.

The top 5 features of NgRx that are often considered the most valuable for state management in Angular applications are:

- **Centralized State Management:** NgRx provides a central store where all the application's data and state are stored. This centralized approach simplifies data access and management, ensuring a single source of truth for your application.
- **Predictable State Changes:** NgRx enforces a unidirectional data flow, making state changes predictable and traceable. Actions and reducers explicitly define how the state evolves, which helps in debugging and maintaining the application**.**
- **Time-Travel Debugging:** NgRx's integration with browser DevTools allows developers to step backward and forward in the application's state history, aiding in debugging and understanding how the application's state evolves over time.
- **Type Safety:** NgRx leverages TypeScript, ensuring strong typing throughout your application. This results in fewer runtime errors and better tooling support, making it easier to work with complex data structures**.**
- **Middleware and Effects:** NgRx supports middleware, enabling you to intercept and process actions before they reach reducers. Effects provide a structured way to handle side effects, such as asynchronous operations, without cluttering your components.

10

How to do Performance Improvement ?

Software developers frequently use Angular to create advanced structurally and content heavy applications. Angular is a powerful framework by nature, but there are occasions when speed optimization may be a problem when it comes to delivering a better user experience.

Here below are some best practice to optimize the Angular application:

- Use trackBy in ngFor loops
- Use lazy loading
- Avoid using ngIf with complex expressions
- Use OnPush change detection strategy
- Use immutable data structures
- Use AOT compilation
- Use Angular Universal for server-side rendering
- Use RxJS for reactive programming
- Use NgRx for state management
- Use Web Workers