**Angular**: Angular is a popular open-source front-end web framework maintained by Google. It is used for building dynamic single-page web applications (SPAs) and supports the development of large-scale, feature-rich applications. Angular provides a structured approach to web development with features like two-way data binding, dependency injection, routing, and more.

**What are the key features of Angular?**

**Angular comes with several key features:**

**Two-way data binding:** It synchronizes the data between the model and the view automatically.

**Dependency Injection:** Angular has a built-in dependency injection mechanism that helps manage the dependencies of components.

**Directives:** Directives are markers on a DOM element that tell Angular to attach a specific behavior to it.

**Routing:** Angular provides a robust routing mechanism for building single-page applications.

**Services:** Services are reusable components that can be injected into Angular components to provide functionality.

**Templates:** Angular uses HTML templates to define the UI of an application, enhancing readability and maintainability.

**Module:**  in Angular refers to a place where you can group the components, directives, pipes, and services, which are related to the application. dedicated to an application domain, workflow, or closely related set of capabilities.

**Data binding:** Data binding in Angular is the automatic synchronization of data between the model (component) and the view (HTML template) in an application. Angular supports both one-way and two-way data binding:

**One-way data binding:** Data flows in one direction from the component to the view (template).

**Two-way data binding**: Data flows in both directions, allowing changes in the view to update the model and vice versa.

**Angular Directive:** Angular directives are markers on a DOM element that tell Angular to attach a specific behavior to it. Directives are used to extend the functionality of HTML elements or to create reusable components. Angular provides three types of directives:

Example:

**Component Directive:** Angular's built-in component directive is exemplified by directives like ngFor and ngIf. These directives define behavior, templates, and styles for reusable components.

**Attribute Directive:** Angular's built-in attribute directives include ngModel, ngStyle, and ngClass, which modify the appearance or behavior of elements by applying styles, classes, or other attributes dynamically.

**Structural Directive:** Angular's built-in structural directives like ngIf, ngFor, and ngSwitch dynamically add or remove elements from the DOM based on conditions, affecting the structure and layout of the rendered HTML.

**Dependency Injection:** Dependency Injection (DI) is a design pattern in which a class's dependencies are provided from the outside. Angular has a built-in dependency injection system that helps manage the dependencies of components, services, and other objects. DI promotes loose coupling, making components easier to test and maintain.

**Angular components:** Angular components are the basic building blocks of Angular applications. A component in Angular is a TypeScript class with a template and a decorator. Components encapsulate the UI and behavior of a part of the application. They consist of a template, which defines the view, and a class, which defines the behavior.

**Angular services:** Angular services are classes that are responsible for encapsulating and abstracting reusable business logic, data access code, or other services. Services in Angular are typically injected into components or other services via Angular's dependency injection system.

**Angular routing:** Angular routing is a mechanism for navigating between different components/views in a single-page application. Angular Router provides a declarative way to define routes and associate them with components. It allows users to navigate by clicking links, entering URLs, or using programmatic navigation.

**Angular CLI**: Angular CLI (Command Line Interface) is a command-line tool provided by the Angular team for building, and maintaining Angular applications. It simplifies the development process by providing commands for generating components, services, modules, and more. Angular CLI also comes with built-in support for testing, linting, and bundling.

**Angular Template Syntax** is a set of rules and conventions used to bind data and define the structure of HTML templates in Angular applications. It includes features like interpolation, property binding, event binding, and template expressions.

**<p>Hello, {{ name }}!</p>**

**<input [value]="name" (input)="name = $event.target.value" />**

**<button (click)="submit()">Submit</button>**

**Angular Forms**: Angular Forms allow you to create and manage user input forms in Angular applications. Angular supports two types of forms: *Template-driven forms* and *Reactive forms*. Template-driven forms are built using directives in the template, while Reactive forms are built programmatically using *form controls* and *form groups*.

**Angular Lifecycle Hooks:**  are methods that allow you to tap into specific moments in the lifecycle of a component or directive in Angular. They provide hooks for initialization, change detection, and destruction of components. Some of the commonly used lifecycle hooks include ngOnInit, ngOnChanges, ngOnDestroy, etc.

**Pipes:** Pipes in Angular are used to transform data without directly impacting the properties holding the data.

We can transform strings, currency amounts, dates, and other data for display. Pipes are simple functions to use in template expressions to accept an input value and return a transformed value.

We can use built-in pipes and can also create custom pipes as well.

Built-in Pipes

Some of Built-in pipes include:

uppercase (to convert string in upper case)

lowercase (to convert string in upper case)

date (to format the date into different types)

Async (subscribes to an Observable or Promise and returns the latest value it has emitted)

Let’s understand them with an example:

<p>{{ userName | uppercase }}</p>

<p>{{ userCountry | lowercase }}</p>

<!-- {{quizData | json}} -->

**How does Angular handle HTTP requests:** Angular provides the HttpClient module for making HTTP requests to a server from an Angular application. HttpClient is available as an injectable service, making it easy to use in components and services. It supports methods like get(), post(), put(), delete(), etc., for making different types of HTTP requests.

Example :

import { HttpClient } from '@angular/common/http';

constructor(private http: HttpClient) {}

getData() {

return this.http.get('https://api.example.com/data');

}

**Angular Guards:** Angular Guards are used to control navigation in Angular applications. They can be used to protect routes, perform authentication, and execute other tasks before allowing a user to navigate to a certain route. Angular provides several types of guards like CanActivate, CanDeactivate, CanLoad, etc.

**Angular Interceptors:** Angular Interceptors are used to intercept HTTP requests and responses *globally* in an Angular application. Interceptors can be used for tasks like adding headers to requests, handling errors, logging, etc. They provide a centralized way to handle HTTP communication logic.