Technical Interview Questions - Angular

1. What are the main building blocks of Angular?

Angular is built on several key components that work together:

Components: These are the basic building blocks in Angular. A component consists of an HTML template, a TypeScript class, and CSS styles. Components are like puzzle pieces that make up the user interface. In my J-Flix project, I created components like movie-card, user-profile, and welcome-page.

Modules: Modules help organize the application into functional units. The main module is AppModule, but you can create feature modules too. They bundle related components, directives, and services together. I found this helpful for organizing different features of my movie app.

Templates: These are HTML files with Angular-specific syntax. They let you create dynamic content using data binding (like {{ movie.Title }}), event binding, and directives.

Directives: Instructions that tell Angular how to manipulate the DOM. There are structural directives like *ngFor (which I used to loop through movies) and attribute directives like ngClass.

Services: Classes that handle data and logic separate from components. Services are great for sharing data between components. In my project, I used a service to fetch data from the movie API.

Dependency Injection: A design pattern used in Angular to provide instances of classes that a component needs. I used this to inject my FetchApiDataService into components.

2. What are the advantages and disadvantages of using Angular?

Advantages:

- **Structure**: Angular enforces a clear structure, which helped me organize my code better. As a beginner, I appreciated having guidelines to follow.
- **TypeScript**: Angular uses TypeScript, which catches errors at compile time rather than runtime. This saved me from many potential bugs.
- **Two-way data binding**: Changes in the UI automatically update the component's data model and vice versa. This made forms much easier to implement.

- **Complete solution**: Angular is a full-fledged framework with everything included - routing, forms, HTTP, testing tools, etc. I didn't need to piece together different libraries.
- **Angular CLI**: The Command Line Interface made it easy to generate components, services, and other elements, saving me time.

Disadvantages:

- **Steep learning curve**: Angular has a lot of concepts to grasp. It took me time to understand terms like dependency injection and observables.
- **Verbosity**: Angular requires more code compared to some other frameworks. Setting up even a simple component requires several files.
- **Size**: Angular applications can be larger than those built with other frameworks, though this is improving in newer versions.
- **Complex for simple applications**: For a small project, Angular might be overkill. It's more suited for larger applications.
- **Frequent updates**: Angular releases major updates every 6 months, which can be hard to keep up with as a student.

3. What is Angular Material and what can you do with it?

Angular Material is a UI component library for Angular that implements Google's Material Design principles. It's like a toolkit of pre-built, styled components that you can use in your Angular application.

With Angular Material, you can:

- **Add professional-looking UI components**: In my J-Flix project, I used MatCard for movie cards, MatDialog for displaying movie details, and MatButtons for actions.
- **Create responsive layouts**: Angular Material includes a flexible grid system that helps create responsive designs that work well on different screen sizes.
- **Implement consistent styling**: All Material components follow the same design language, so they look and feel consistent.
- **Use pre-built animations**: Material comes with animations for things like opening dialogs or expanding panels, which saved me from writing complex animation code.
- **Customize themes**: You can customize colors, typography, and other aspects to match your brand or design preferences.

I found Angular Material particularly helpful because it let me focus on functionality rather than spending too much time styling components from scratch. The documentation is also really good, with plenty of examples that helped me implement features quickly.