Technical Interview Questions - Angular

1. Briefly explain what Angular is and its various uses.

From what I've learned, Angular is a comprehensive front-end framework developed by Google that's built with TypeScript. It's designed for creating single-page applications where you want a smooth, app-like experience for users.

What stands out to me about Angular is its component-based architecture. You essentially build your application by creating different components that handle specific parts of your UI and functionality, which makes the code more organized and reusable.

Based on my studies and course projects, Angular can be used for:

- Building complex web applications where you need structure and organization. The framework gives you tools for managing everything from routing to forms in a consistent way.
- Creating Progressive Web Apps that can work offline and provide a more native-like experience than traditional websites.
- Developing cross-platform applications. From what I understand, you can use your Angular knowledge with frameworks like Ionic to build mobile apps too.
- Data-driven applications like dashboards or admin panels where you're continuously updating content based on user interactions or backend data.

I like that Angular provides a complete solution right out of the box. In my personal projects with other technologies, I've often had to piece together different libraries, but Angular seems to include most of what you need from the start.

2. Describe the key differences between TypeScript and JavaScript.

From my perspective as someone who's used both in course projects and personal work, TypeScript is essentially JavaScript with added type checking and some extra features.

The main difference I've noticed is that TypeScript uses static typing. When I write code in TypeScript, I can specify exactly what type each variable should be, what parameters functions should accept, and what they should return. This has helped me catch errors during development that I would have only discovered at runtime with regular JavaScript.

Another key difference is that TypeScript code needs to be compiled down to JavaScript before it can run in a browser. In my projects, I write in TypeScript but what actually executes is the JavaScript it produces.

I've found that TypeScript gives you enhanced object-oriented programming features compared to vanilla JavaScript. Things like interfaces, which act as blueprints for objects, and access modifiers like private and public, which help control how properties can be accessed. These concepts were familiar to me from learning other programming languages.

The tooling support is also different. When I've used TypeScript in VS Code, the editor can provide much better suggestions and catch errors as I type, which has been really helpful while learning.

Even though I'm still learning both languages, I can see how TypeScript adds structure that becomes increasingly valuable as projects get larger and more complex.

3. What are some advantages of using TypeScript?

Based on my experience using TypeScript in personal projects and course assignments, I've noticed several advantages over plain JavaScript.

The biggest benefit for me has been catching errors early. When I'm writing code, TypeScript immediately highlights issues like trying to use a string method on a number, or forgetting to pass a required parameter to a function. This has saved me a lot of debugging time compared to my JavaScript projects.

I also really appreciate the improved code completion in my editor. When I'm working with an object in TypeScript, my editor shows me exactly what properties and methods are available, which is super helpful when learning new libraries or APIs.

The self-documenting nature of TypeScript has been great too. When I look at functions I wrote weeks ago for a project, I can immediately see what kinds of parameters they expect and what they return, without having to read through all the code again.

For learning purposes, I've found that TypeScript actually helped me understand JavaScript better. Having to think about what types my variables and functions use has made me more aware of how JavaScript works under the hood.

Even in my small projects, I've noticed that adding type definitions makes my code more robust. When I needed to modify functionality in a TypeScript project, the compiler guided me through all the places I needed to update, which prevented potential bugs.

I like that TypeScript is gradually adoptable too. In one of my projects, I started with minimal typing and added more detailed types as I went along, which made the learning curve more manageable.