Comprehension Questions

How does TypeScript's class system enable you to create blueprints for objects with specific properties and behaviors?

How do conditional types allow TypeScript types to adapt based on other types?

How do advanced types (union, intersection, mapped, conditional, utility) contribute to building flexible and type-safe applications?

What is a mapped type, and how does it help in transforming or generating new types from existing ones?

What is the difference between `protected` and `private` access modifiers in terms of class inheritance and member accessibility

What are utility types in TypeScript, and why are they important for rapid development and code maintainability?

How does TypeScript allow you to declare and initialize class properties directly in the constructor using access modifiers

Explain the difference between writing a function with any type and using a generic type parameter. What are the implications for type safety and code maintainability?

Why is using generics preferable to duplicating classes for different data types, as shown in the FeedbackBox example? What problems does this approach solve?

How do generics help maintain type safety when building reusable components for a platform like EduFlow, which handles diverse content types?

Describe a workflow for refactoring repeated code to use generics. What steps should a programmer follow to ensure flexibility and type safety?